ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΣΗΜΕΙΩΣΕΙΣ
ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΣΥΣΤΗΜΑΤΟΣ

ΠΑΝΑΓΙΩΤΗΣ ΣΤΑΜΑΤΟΠΟΥΛΟΣ

ΑΘΗΝΑ – 2007

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ

Περιεχόμενο του μαθήματος

- Το λειτουργικό σύστημα Unix από την πλευρά του χρήστη
- Βοηθητικά προγράμματα του Unix
- Προγραμματισμός σε κελύφη
- Προγραμματισμός λειτουργιών συστήματος σε C για
  - χειρισμό λαθών
  - δημιουργία και τερματισμό διεργασιών
  - αποστολή/παραλαβή σημάτων
  - είσοδο/έξοδο χαμηλού επιπέδου
  - επικοινωνία μεταξύ διεργασιών μέσω σωλήνων, υποδοχών, ουρών μηνυμάτων, κοινής μνήμης και σηματοφόρων
  - δημιουργία, τερματισμό και συγχρονισμό νημάτων
  - διαχείριση συστήματος αρχείων

Λειτουργικό σύστημα

- Ενδιάμεσο πρόγραμμα μεταξύ του χρήστη και του υλικού ενός υπολογιστή
- Στόχοι
  - Διευκόλυνση της χρήσης του υπολογιστή
  - Αποδοτική χρήση του υπολογιστή
- Διαχείριση πόρων
  - Κεντρική μονάδα επεξεργασίας
  - Κύρια μνήμη
  - Δευτερεύουσα μνήμη
  - Συσκευές εισόδου/εξόδου

Unix

- Στην πρωταρχική του μορφή κατασκευάστηκε από τον Ken Thompson το 1969 σε γλώσσα assembly για ένα PDP-7 στα Bell Laboratories της AT&T
- Μεταφέρθηκε σε ένα PDP-11 το 1971 από τον Ken Thompson και τον Dennis Ritchie
- Το 1973 υλοποιήθηκε εξ αρχής στο μεγαλύτερο μέρος του στη γλώσσα C, η οποία αναπτύχθηκε από τον Brian Kernighan και τον Dennis Ritchie
- Πολλές βελτιώσεις έγιναν από το University of California, Berkeley (UCB)
- Ενώ για πολύ μεγάλο διάστημα, περίπου μία 25ετία, το Unix ήταν περιορισμένο σε ακαδημαϊκά και ερευνητικά περιβάλλοντα μόνο, σήμερα, ως Linux, έχει "καταλάβει" τους προσωπικούς υπολογιστές πολλών χρηστών
- Εδώ και μερικά χρόνια είναι σε εξέλιξη μία διαρκής προσπάθεια τυποποίησής του (IEEE POSIX, Open Group)

Βιβλίο "αναφοράς"

Για τη συνέχεια, το [§$x.y$–**KP**] θα αναφέρεται στην παράγραφο $x.y$ του βιβλίου:

> Brian W. Kernighan, Rob Pike, "Το Περιβάλλον Προγραμματισμού UNIX", Κλειδάριθμος, 1989.

Γενικά χαρακτηριστικά

- Λειτουργικό σύστημα πολλών χρηστών
- Υποστήριξη πολλαπλών διεργασιών με διαμέριση χρόνου
- Παροχή ασφάλειας
- Υποστήριξη δικτύων
- Δομή του Unix
  - Πυρήνας
  - Βοηθητικά προγράμματα
  - Κέλυφος (C, Bourne, Korn, ...)
- Προγραμματισμός σε C μέσω κλήσεων συστήματος
- Διαλογική επικοινωνία με το χρήστη μέσω του κελύφους
- Προγραμματισμός του κελύφους
- Διάκριση μεταξύ κεφαλαίων και πεζών

Ιεραρχικό σύστημα αρχείων

- Κατάλογος-ρίζα (/)
- Τρέχων κατάλογος (.)
- Γονικός κατάλογος (..)
- Ονόματα-μονοπάτια
  - Απόλυτα
  - Σχετικά
- Αρχεία
- Σύνδεσμοι
- Συσκευές

Λογαριασμός χρήστη

- Ονομα χρήστη
- Συνθηματικό
- Κέλυφος αρχικής σύνδεσης
- Αρχική ομάδα
- Κατάλογος αφετηρίας

Αρχική σύνδεση

```
Red Hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on an i586
login: spro
Password:
Last login: Sun Jan 27 12:31:00 from kronos.di.uoa.gr
$
```

Συχνότερα χρησιμοποιούμενες εντολές

`man` : Εμφάνιση οδηγιών χρήσης εντολών
`ls` : Εμφάνιση περιεχομένων καταλόγου
`pwd` : Εμφάνιση τρέχοντος καταλόγου
`cd` : Αλλαγή τρέχοντος καταλόγου
`mkdir` : Δημιουργία καταλόγων
`rmdir` : Διαγραφή καταλόγων
`cp` : Αντιγραφή αρχείων
`mv` : Μετακίνηση ή μετονομασία αρχείων
`rm` : Διαγραφή αρχείων
`cat` : Εμφάνιση αρχείων
`lpr` : Εκτύπωση αρχείων
`vi` : Κειμενογράφος οθόνης

Εντολή man (επιλογή -k)

```
$ man pwd
PWD(1)                          FSF                             PWD(1)

NAME
       pwd - print name of current/working directory

SYNOPSIS
       pwd [OPTION]

DESCRIPTION
       Print  the full filename of the current working directory.

       --help display this help and exit

       --version
              output version information and exit

AUTHOR
       Written by Jim Meyering.

REPORTING BUGS
       Report bugs to <bug-sh-utils@gnu.org>.

COPYRIGHT
$ man -k manual
man                   (1)  - format and display the on-line manual
                             pages
man [manpath]         (1)  - format and display the on-line manual
                             pages
man2html              (1)  - format a manual page in html
perlxs                (1)  - XS language reference manual
wget                  (1)  - GNU Wget Manual
whereis               (1)  - locate the binary, source, and manual page
                             files for a command
xman                  (1x)  - Manual page display program for the X
                              Window System
$
```

Εντολή `ls` (επιλογές `-a`, `-l`, `-r`)

```
$ ls
bin
$ ls -a
.   .bash_history  .bash_profile  bin     .screenrc
..  .bash_logout   .bashrc        .emacs
$ ls -al
total 40
drwx------    4 spro     users        4096 Jan 27 13:13 .
drwxr-xr-x    7 root     root         4096 Jan 27 12:10 ..
-rw-------    1 spro     users         358 Jan 27 12:33 .bash_history
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
$ ls -al .bashrc
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
$ ls -alr
total 40
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-------    1 spro     users         358 Jan 27 12:33 .bash_history
drwxr-xr-x    7 root     root         4096 Jan 27 12:10 ..
drwx------    4 spro     users        4096 Jan 27 13:13 .
$
```

- Δικαιώματα προστασίας (εξουσιοδοτήσεις) [§2.4–**KP**]
  - Δικαιώματα ιδιοκτήτη (στήλες 2-4)
  - Δικαιώματα ομάδας (στήλες 5-7)
  - Δικαιώματα άλλων (στήλες 8-10)

Εντολή `ls` (επιλογές `-d`, `-F`, `-R`, `-t`)

```
$ ls -al
total 40
drwx------    4 spro     users        4096 Jan 27 13:20 .
drwxr-xr-x    7 root     root         4096 Jan 27 12:10 ..
-rw-------    1 spro     users        2232 Jan 27 17:40 .bash_history
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
$ ls -al bin
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 .
drwx------    4 spro     users        4096 Jan 27 13:20 ..
$ ls -ald bin
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
$ ls -aFl
total 40
drwx------    4 spro     users        4096 Jan 27 13:20 ./
drwxr-xr-x    7 root     root         4096 Jan 27 12:10 ../
-rw-------    1 spro     users        2232 Jan 27 17:40 .bash_history
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin/
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
$ ls -aR .
.:
.   .bash_history  .bash_profile  bin     .screenrc
..  .bash_logout   .bashrc        .emacs

./bin:
.  ..
$ ls -alt
total 40
-rw-------    1 spro     users        2232 Jan 27 17:40 .bash_history
drwx------    4 spro     users        4096 Jan 27 13:20 .
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    7 root     root         4096 Jan 27 12:10 ..
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
$
```

Εντολές pwd, cd, mkdir, rmdir

```
$ pwd
/home/spro
$ ls -a
.   .bash_history  .bash_profile  bin     .screenrc
..  .bash_logout   .bashrc        .emacs
$ mkdir subdir
$ ls -lF
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin/
drwxr-xr-x    2 spro     users        4096 Jan 27 22:14 subdir/
$ cd subdir
$ ls -al
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 22:14 .
drwx------    5 spro     users        4096 Jan 27 22:14 ..
$ pwd
/home/spro/subdir
$ cd ..
$ ls -aR
.:
.   .bash_history  .bash_profile  bin     .screenrc
..  .bash_logout   .bashrc        .emacs  subdir

./bin:
.  ..

./subdir:
.  ..
$ rmdir subdir
$ ls subdir
ls: subdir: No such file or directory
$
```

Εντολές `cp` (επιλογές `-i`, `-r`), `mv` (επιλογή `-i`), `rm` (επιλογές `-i`, `-r`)

```
$ ls -a
.   .bash_history  .bash_profile  bin     .screenrc
..  .bash_logout   .bashrc        .emacs
$ cp .bash_profile newfile
$ ls -a
.   .bash_history  .bash_profile  bin     newfile
..  .bash_logout   .bashrc        .emacs  .screenrc
$ mv newfile anotherfile
$ ls -a
.   anotherfile    .bash_logout   .bashrc  .emacs
..  .bash_history  .bash_profile  bin      .screenrc
$ cp -i .bashrc anotherfile
cp: overwrite ‘anotherfile’? n
$ cp /bin/true .
$ mkdir direct
$ cp /bin/false direct
$ mv -i true direct/false
mv: overwrite ‘direct/false’? y
$ cp -r direct directory
$ mv anotherfile directory
$ ls -aR
.:
.   .bash_history  .bash_profile  bin     directory  .screenrc
..  .bash_logout   .bashrc        direct  .emacs

./bin:
.  ..

./direct:
.  ..  false

./directory:
.  ..  anotherfile  false
$ rm directory/anotherfile
$ rm -r direct
$ rm -ri directory
rm: descend into directory ‘directory’? y
rm: remove ‘directory/false’? y
rm: remove directory ‘directory’? y
$ ls -a
.   .bash_history  .bash_profile  bin     .screenrc
..  .bash_logout   .bashrc        .emacs
$
```

Εντολές cat (επιλογή -n), lpr (επιλογές -h, -P)

```
$ cat .bashrc
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

PS1='$ '
$ cat -n .bashrc
     1  # .bashrc
     2
     3  # User specific aliases and functions
     4
     5  # Source global definitions
     6  if [ -f /etc/bashrc ]; then
     7          . /etc/bashrc
     8  fi
     9
    10  PS1='$ '
$ lpr /etc/passwd
$ lpr -h -Plexmark ../../etc/group
$
```

Κειμενογράφος οθόνης `vi`

- Σύνταξη: `vi` $\langle filename \rangle$
- Έλεγχος οθόνης και δρομέα
  $\leftarrow$ | $\downarrow$ | $\uparrow$ | $\rightarrow$: Μετακίνηση αριστερά | κάτω | επάνω | δεξιά κατά ένα χαρακτήρα
  `[Back Space]` | `[Return]` | `-` | `[Space]` : Όπως προηγουμένως
  `h` | `j` | `k` | `l` : Όπως προηγουμένως
  `0` | `$` : Μετακίνηση στην αρχή | τέλος της τρέχουσας γραμμής
  `H` | `L` | `M` : Μετακίνηση στην πρώτη | τελευταία | μεσαία γραμμή της οθόνης
  `G` | $\langle n \rangle$`G` : Μετακίνηση στην τελευταία | $\langle n \rangle$-οστή γραμμή του αρχείου
  `^F` | `^B` : Μετακίνηση κάτω | επάνω μία οθόνη
  `^D` | `^U` : Μετακίνηση κάτω | επάνω μισή οθόνη
  `^E` | `^Y` : Εμφάνιση μίας επιπλέον γραμμής στο τέλος | αρχή της οθόνης
  `^G` : Εμφάνιση του αριθμού της τρέχουσας γραμμής
  `^L` : Επανασχεδίαση της οθόνης
- Εντολές διόρθωσης
  `i`$\langle t \rangle$`[Esc]` | `a`$\langle t \rangle$`[Esc]` : Εισαγωγή του κειμένου $\langle t \rangle$ πριν | μετά το δρομέα
  `I`$\langle t \rangle$`[Esc]` | `A`$\langle t \rangle$`[Esc]` : Εισαγωγή του κειμένου $\langle t \rangle$ στην αρχή | τέλος της τρέχουσας γραμμής

o⟨t⟩[Esc] | O⟨t⟩[Esc] : Δημιουργία μίας κενής γραμμής μετά | πριν την τρέχουσα γραμμή και εισαγωγή του κειμένου ⟨t⟩

x | ⟨n⟩x : Διαγραφή ενός | ⟨n⟩ χαρακτήρων από τη θέση του δρομέα και μετά

Y | ⟨n⟩Y : Φύλαξη μίας | ⟨n⟩ γραμμών από την τρέχουσα γραμμή και μετά

dd | ⟨n⟩dd : Διαγραφή μίας | ⟨n⟩ γραμμών από την τρέχουσα γραμμή και μετά

p | P : Εισαγωγή των γραμμών που φυλάχθηκαν ή διαγράφηκαν πιο πρόσφατα μετά | πριν την τρέχουσα γραμμή

cw⟨t⟩[Esc] : Αντικατάσταση των χαρακτήρων μέχρι και το τέλος της τρέχουσας λέξης με το κείμενο ⟨t⟩

dw : Διαγραφή χαρακτήρων μέχρι και το τέλος της τρέχουσας λέξης

C⟨t⟩[Esc] : Αντικατάσταση των χαρακτήρων μέχρι και το τέλος της τρέχουσας γραμμής με το κείμενο ⟨t⟩

D : Διαγραφή των χαρακτήρων μέχρι και το τέλος της τρέχουσας γραμμής

r⟨c⟩ : Αντικατάσταση του τρέχοντα χαρακτήρα με το χαρακτήρα ⟨c⟩

R⟨t⟩[Esc] : Αντικατάσταση των χαρακτήρων πλήθους όσο το μήκος του κειμένου ⟨t⟩ με το ⟨t⟩

~ : Αλλαγή από μικρό σε κεφαλαίο και αντίστροφα του τρέχοντα χαρακτήρα

J : Συνένωση της τρέχουσας και της επόμενης γραμμής

. : Επανάληψη της πιο πρόσφατης διόρθωσης

u : Ακύρωση της πιο πρόσφατης διόρθωσης

ZZ : Φύλαξη των αλλαγών στο αρχείο και έξοδος

- Εντολές αναζήτησης

  /⟨s⟩[Return] | ?⟨s⟩[Return] : Αναζήτηση προς τα εμπρός | πίσω της συμβολοσειράς ⟨s⟩ (η . στο ⟨s⟩ ερμηνεύεται σαν "οποιοσδήποτε χαρακτήρας", το .* ταιριάζει με μηδέν ή περισσότερους χαρακτήρες, το ^ και το $ υποδηλώνουν την αρχή και το τέλος μίας γραμμής, αντίστοιχα)

  n | N : Επανάληψη της αναζήτησης προς την ίδια | αντίθετη κατεύθυνση

- Εντολές του κειμενογράφου γραμμής ed

  :⟨n⟩$_1$,⟨n⟩$_2$d[Return] : Διαγραφή των γραμμών από τη ⟨n⟩$_1$ μέχρι και τη ⟨n⟩$_2$ (εκτός από αριθμοί, τα ⟨n⟩$_i$ μπορεί να είναι ., που δηλώνει την τρέχουσα γραμμή, ή $, που δηλώνει την τελευταία γραμμή

  :⟨n⟩$_1$,⟨n⟩$_2$s/⟨s⟩$_1$/⟨s⟩$_2$/g[Return] : Αντικατάσταση της συμβολοσειράς ⟨s⟩$_1$ με τη συμβολοσειρά ⟨s⟩$_2$ από τη γραμμή ⟨n⟩$_1$ μέχρι και τη ⟨n⟩$_2$ (για τις συμβολοσειρές ⟨s⟩$_i$, ισχύει η ίδια σύμβαση όπως και στις εντολές αναζήτησης)

  :r ⟨f⟩[Return] : Εισαγωγή του αρχείου ⟨f⟩ μετά την τρέχουσα γραμμή

  :w[Return] : Φύλαξη των αλλαγών στο αρχείο

  :q![Return] : Έξοδος χωρίς φύλαξη των αλλαγών στο αρχείο

  :wq[Return] | :x[Return] : Φύλαξη των αλλαγών στο αρχείο και έξοδος (όπως και η εντολή ZZ)

Αλλες εντολές του Unix

`chmod` : Αλλαγή δικαιωμάτων προστασίας αρχείων και καταλόγων

`more` : Εμφάνιση αρχείων σελίδα-σελίδα

`grep` : Αναζήτηση συμβολοσειρών σε αρχεία

`wc` : Εμφάνιση αριθμού γραμμών, λέξεων και χαρακτήρων σε αρχεία

`sort` : Ταξινόμηση αρχείων

`touch` : Δημιουργία κενού αρχείου

`ln` : Δημιουργία σκληρών ή συμβολικών συνδέσμων (περί inodes, δείτε [§2.5–KP])

`echo` : Εμφάνιση μηνυμάτων

`date` : Εμφάνιση ημερομηνίας και ώρας

`passwd` : Αλλαγή συνθηματικού

`hostname` : Εμφάνιση ονόματος μηχανήματος

`whoami` : Εμφάνιση ονόματος χρήστη

`lpq` : Εμφάνιση ουράς εκτυπωτή

`lprm` : Ακύρωση εκτύπωσης από ουρά εκτυπωτή

`cmp` : Σύγκριση δύο αρχείων

`diff` : Εμφάνιση διαφορών δύο αρχείων κειμένου

`head` : Εμφάνιση αρχικών γραμμών αρχείων

`tail` : Εμφάνιση τελικών γραμμών αρχείων

Εντολή chmod (επιλογή -R)

```
$ cp .bash_profile tmpfile
$ ls -l
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         191 Feb  2 20:08 tmpfile
$ chmod 744 tmpfile
$ ls -l
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rwxr--r--    1 spro     users         191 Feb  2 20:08 tmpfile
$ chmod o-r tmpfile
$ chmod g+wx tmpfile
$ ls -l
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rwxrwx---    1 spro     users         191 Feb  2 20:08 tmpfile
$ chmod u-wx,g=rx,o+r tmpfile
$ ls -l
total 8
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-r--r-xr--    1 spro     users         191 Feb  2 20:08 tmpfile
$ rm tmpfile
rm: remove write-protected file ‘tmpfile’? y
$ cp -r /etc/cipe .
$ ls -l cipe
total 8
-rwxr-xr-x    1 spro     users         620 Feb  2 20:10 ip-down
-rwxr-xr-x    1 spro     users        1632 Feb  2 20:10 ip-up
$ chmod -R 750 cipe
$ ls -lRF cipe
cipe:
total 8
-rwxr-x---    1 spro     users         620 Feb  2 20:10 ip-down*
-rwxr-x---    1 spro     users        1632 Feb  2 20:10 ip-up*
$ rm -r cipe
$
```

Εντολή more

```
$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/dev/null
rpm:x:37:37::/var/lib/rpm:/bin/bash
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
ntp:x:38:38::/etc/ntp:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/bin/false
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
--More--(74%)
```

- Δυνατότητες

  [Return] : Εμφάνιση μίας επιπλέον γραμμής

  [Space] : Εμφάνιση μίας επιπλέον σελίδας

  b : Εμφάνιση προηγούμενης σελίδας

  /⟨s⟩[Return] : Αναζήτηση προς τα εμπρός της συμβολοσειράς ⟨s⟩

  !⟨c⟩[Return] : Εκτέλεση της εντολής ⟨c⟩

  v : Κλήση του κειμενογράφου vi

  . : Επανάληψη της προηγούμενης εντολής

  h : Περιγραφή των δυνατοτήτων της more

  q : Έξοδος από τη more

Εντολές grep (επιλογές -n, -i, -v), wc (επιλογές -l, -w, -c)

```
$ grep 6000 /etc/login.defs
UID_MAX                 60000
GID_MAX                 60000
$ grep -n rc .bashrc
1:# .bashrc
5:# Source global definitions
6:if [ -f /etc/bashrc ]; then
7:      . /etc/bashrc
$ grep -i RoOt /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
$ grep -v '#' /etc/hosts.allow
ALL: 195.134.65.
ALL: 195.134.66.
ALL: 195.134.67.
ALL: 195.134.68.
ALL: 195.134.69.
$ wc .bash_profile
     13      29     191 .bash_profile
$ wc -l .bash_profile
     13 .bash_profile
$ wc -w .bash_profile
     29 .bash_profile
$ wc -c .bash_profile
    191 .bash_profile
$ wc -cl .bash_profile
     13     191 .bash_profile
$ ls -al .bash_profile
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
$
```

Εντολή `sort` (επιλογές `+`$\langle n \rangle$, `-r`, `-n`)

```
$ cat example.txt
john          32          london
jean          7           paris
marco         21          rome
anna          18          athens
antonio       58          madrid
peter         5           chicago
ahmet         27          ankara
luis          1           geneva
ji            41          peking
filip         18          brussels
$ sort example.txt
ahmet         27          ankara
anna          18          athens
antonio       58          madrid
filip         18          brussels
jean          7           paris
ji            41          peking
john          32          london
luis          1           geneva
marco         21          rome
peter         5           chicago
$ sort +1 example.txt
anna          18          athens
filip         18          brussels
luis          1           geneva
marco         21          rome
ahmet         27          ankara
john          32          london
ji            41          peking
antonio       58          madrid
peter         5           chicago
jean          7           paris
$ sort -r -n +1 example.txt
antonio       58          madrid
ji            41          peking
john          32          london
ahmet         27          ankara
marco         21          rome
filip         18          brussels
anna          18          athens
jean          7           paris
peter         5           chicago
luis          1           geneva
$
```

Εντολές touch, ln (επιλογή -s)

```
$ touch empty
$ ls -al
total 40
drwx------    3 spro     users        4096 Feb  2 21:51 .
drwxr-xr-x    8 root     root         4096 Feb  2 09:15 ..
-rw-------    1 spro     users        5013 Feb  2 20:11 .bash_history
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    1 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    1 spro     users           0 Feb  2 21:51 empty
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
$ rm empty
$ ln .bashrc hard-link
$ ls -al
total 44
drwx------    3 spro     users        4096 Feb  2 21:52 .
drwxr-xr-x    8 root     root         4096 Feb  2 09:15 ..
-rw-------    1 spro     users        5013 Feb  2 20:11 .bash_history
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    2 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    2 spro     users         134 Jan 27 12:24 hard-link
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
$ ln -s /etc/group symb-link
$ ls -al
total 44
drwx------    3 spro     users        4096 Feb  2 21:52 .
drwxr-xr-x    8 root     root         4096 Feb  2 09:15 ..
-rw-------    1 spro     users        5013 Feb  2 20:11 .bash_history
-rw-r--r--    1 spro     users          24 Jan 27 12:10 .bash_logout
-rw-r--r--    1 spro     users         191 Jan 27 12:10 .bash_profile
-rw-r--r--    2 spro     users         134 Jan 27 12:24 .bashrc
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users         820 Jan 27 12:10 .emacs
-rw-r--r--    2 spro     users         134 Jan 27 12:24 hard-link
-rw-r--r--    1 spro     users        3511 Jan 27 12:10 .screenrc
lrwxrwxrwx    1 spro     users          10 Feb  2 21:52 symb-link ->
                                                        /etc/group
$ ls -F
bin/  hard-link  symb-link@
$ rm hard-link symb-link
$
```

Εντολές echo (επιλογή -n),
date, passwd, hostname, whoami,
lpq (επιλογή -P),
lprm (επιλογή -P)

```
$ echo This is a message produced by echo
This is a message produced by echo
$ echo -n This is a message produced by echo
This is a message produced by echo$
$ date
Sat Feb  2 21:58:13 EET 2002
$ passwd
Changing password for spro
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully
$ hostname
galini
$ whoami
spro
$ lpr /etc/sendmail.cf
$ lpq
lp is ready and printing
Rank   Owner      Job  Files                                 Total Size
active spro       143  /etc/sendmail.cf                      46365 bytes
$ lprm 143
dfA143galini dequeued
cfA143galini dequeued
$ lpr -Plp2 /etc/termcap
$ lpr -Plp2 /usr/share/magic
$ lpq -Plp2
lp2 is ready and printing
Rank   Owner      Job  Files                                 Total Size
active spro       144  /etc/termcap                          737535 bytes
1st    spro       145  /usr/share/magic                      226045 bytes
$ lprm -Plp2 spro
dfA144galini dequeued
cfA144galini dequeued
dfA145galini dequeued
cfA145galini dequeued
$
```

Εντολές `cmp`, `diff`,
`head` (επιλογή -$\langle n \rangle$),
`tail` (επιλογή -$\langle n \rangle$)

```
$ cp .bashrc .bashrc.dupl
$ cmp .bashrc .bashrc.dupl
$ rm .bashrc.dupl
$ cmp example1.txt example2.txt
example1.txt example2.txt differ: char 140, line 5
$ diff example1.txt example2.txt
5c5
< antonio       58         madrid
---
> antonio       68         madrid
$ head /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
$ head -2 /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
$ tail /etc/group
mailnull:x:47:
rpm:x:37:
xfs:x:43:
ntp:x:38:
rpc:x:32:
rpcuser:x:29:
nfsnobody:x:65534:
nscd:x:28:
ident:x:98:
radvd:x:75:
$ tail -1 /etc/group
radvd:x:75:
$
```

Κέλυφος C (`csh` ή `tcsh`)

- Είναι μερικές φορές το κέλυφος αρχικής σύνδεσης (εκτός από συστήματα Linux, όπου συνήθως είναι το `bash`)
- Η εντολή `logout` διακόπτει την αρχική σύνδεση
- Η εντολή `csh` (ή `tcsh`) δημιουργεί ένα νέο κέλυφος C
- Η εντολή `exit` τερματίζει ένα κέλυφος C
- Το ~ συμβολίζει τον κατάλογο αφετηρίας
- Το αρχείο `~/.cshrc` περιέχει εντολές που εκτελούνται κατά την ενεργοποίηση ενός κελύφους C
- Το αρχείο `~/.login` περιέχει εντολές που εκτελούνται κατά την αρχική σύνδεση
- Το αρχείο `~/.logout` περιέχει εντολές που εκτελούνται όταν διακόπτεται η αρχική σύνδεση

- Προκαθορισμένη είσοδος stdin (πληκτρολόγιο)
- Προκαθορισμένη έξοδος stdout (οθόνη)
- Προκαθορισμένη έξοδος για διαγνωστικά μηνύματα stderr (οθόνη)
- Για ορισμένες εντολές του `Unix` (π.χ. `cat`, `lpr`, `grep`, `wc`, `sort`, `head`, `tail` κ.λ.π.) τα ορίσματα που είναι ονόματα αρχείων είναι προαιρετικά, υπό την έννοια ότι, αν παραλείπονται, χρησιμοποιείται η προκαθορισμένη είσοδος (το τέλος του stdin δίνεται από το πληκτρολόγιο με το χαρακτήρα `^D`)

```
% wc
This is an example where the command "wc"
is used without filename argument. So, it
counts the lines, words and characters of
the standard input.
^D
      4      25     146
%
```

- Ανακατεύθυνση του stdin (<)

```
% grep reg < .screenrc
# Prepend/append register [/] to the paste if ^a^] is pressed.
register [ "\033:se noai\015a"
register ] "\033:se ai\015a"
%
```

- Ανακατεύθυνση των stdout και stderr (>, >!, >&, >&!)

```
% date > a_file
% cat < a_file
Sun Feb  3 13:54:48 EET 2002
% ls -a > a_file
a_file: File exists.
% echo A test line >! a_file
% cat < a_file
A test line
% cp
cp: missing file arguments
Try ‘cp --help’ for more information.
% cp >& b_file
% cat < b_file
cp: missing file arguments
Try ‘cp --help’ for more information.
% cat c_file >& b_file
b_file: File exists.
% cat c_file >&! b_file
% cat b_file
cat: c_file: No such file or directory
%
```

- Ανακατεύθυνση των stdout και stderr με προσάρτηση (>>, >>!, >>&, >>&!)

```
% cat a_file
A test line
% cat b_file
cat: c_file: No such file or directory
% date >> c_file
c_file: No such file or directory.
% date >> a_file
% cat a_file
A test line
Sun Feb  3 13:59:33 EET 2002
% wc < .bashrc >>! c_file
% cat c_file
      8      21     124
% lls >>& b_file
% cat b_file
cat: c_file: No such file or directory
lls: Command not found.
% cat b_file c_file e_file >>&! d_file
% cat d_file
cat: c_file: No such file or directory
lls: Command not found.
      8      21     124
cat: e_file: No such file or directory
% rm a_file b_file c_file d_file
%
```

- Φίλτρα και σωληνώσεις (|)

```
% ls -al | grep 6
total 36
drwx------    3 sprocsh  users        4096 Feb  3 14:01 .
drwxr-xr-x    8 root     root         4096 Feb  2 09:15 ..
drwxr-xr-x    3 sprocsh  users        4096 Feb  2 09:15 .kde
% cat .bashrc | tail -5 | sort | lpr -h -Plp2
%
```

- Ακολουθίες και ομάδες εντολών (;)

```
% pwd
/home/sprocsh
% mkdir mydir ; cd mydir ; ls -al
total 8
drwxr-xr-x    2 sprocsh  users        4096 Feb  3 14:04 .
drwx------    4 sprocsh  users        4096 Feb  3 14:04 ..
% pwd
/home/sprocsh/mydir
% cd ..
% rmdir mydir
% (mkdir mydir ; cd mydir ; ls -al)
total 8
drwxr-xr-x    2 sprocsh  users        4096 Feb  3 14:05 .
drwx------    4 sprocsh  users        4096 Feb  3 14:05 ..
% pwd
/home/sprocsh
% rmdir mydir
%
```

- Δικαιώματα προστασίας αρχείων και καταλόγων κατά τη δημιουργία τους (umask)

```
% umask
22
% touch file1
% mkdir dir1
% ls -ld file1 dir1
drwxr-xr-x    2 sprocsh  users        4096 Feb  3 14:06 dir1
-rw-r--r--    1 sprocsh  users           0 Feb  3 14:06 file1
% umask 077
% touch file2
% mkdir dir2
% ls -ld file2 dir2
drwx------    2 sprocsh  users        4096 Feb  3 14:07 dir2
-rw-------    1 sprocsh  users           0 Feb  3 14:07 file2
% umask 022
% rm file1 file2
% rmdir dir1 dir2
%
```

- Στοίβες καταλόγων (pushd, popd)

```
% pwd
/home/sprocsh
% pushd /etc
/etc ~
% pwd
/etc
% ls -l passwd
-rw-r--r--    1 root     root          1229 Feb  2 22:02 passwd
% pushd /usr/local
/usr/local /etc ~
% pwd
/usr/local
% ls
bin  doc  etc  games  include  lib  libexec  sbin  share  src
% pushd /usr/bin
/usr/bin /usr/local /etc ~
% pwd
/usr/bin
% ls -l | wc -l
   1402
% popd
/usr/local /etc ~
% pwd
/usr/local
% popd
/etc ~
% pwd
/etc
% ls -l group
-rw-r--r--    1 root     root           483 Dec 27 13:09 group
% popd
~
% pwd
/home/sprocsh
% popd
popd: Directory stack empty.
%
```

- Μεταχαρακτήρες για ονόματα αρχείων

  * : Ταιριάζει με κανένα ή περισσότερους χαρακτήρες

  ? : Ταιριάζει με έναν ακριβώς χαρακτήρα

  $[\langle c\rangle_1\langle c\rangle_2 \ldots \langle c\rangle_n]$: Ταιριάζει με έναν ακριβώς χαρακτήρα από τους $\langle c\rangle_1$, $\langle c\rangle_2$, ... , $\langle c\rangle_n$

  $[\langle c\rangle_1 - \langle c\rangle_2]$: Ταιριάζει με έναν ακριβώς χαρακτήρα μεταξύ των $\langle c\rangle_1$ και $\langle c\rangle_2$ συμπεριλαμβανομένων

```
% pushd /etc
/etc ~
% ls -dF ???
gtk/  kde/  nmh/  ntp/  opt/  ppp/  rmt@  rpc  rpm/  ssh/  X11/
% cp *tab* ~
% ls -dF [bmpwz][acjp]*
bashrc   mail.rc     pam.d/        passwd   warnquota.conf
mail/    makedev.d/  pam_smb.conf  passwd-
mailcap  man.config  paper.config  ppp/
% ls -ldF *[a-e][f-i][k-q][r-z]*
drwxr-xr-x    2 root     root         4096 Dec 27 14:35 cron.daily/
% popd
~
% ls
anacrontab  crontab  fstab  fstab.REVOKE  inittab  mtab
% rm -i *tab*
rm: remove ‘anacrontab’? y
rm: remove ‘crontab’? y
rm: remove ‘fstab’? y
rm: remove ‘fstab.REVOKE’? y
rm: remove ‘inittab’? y
rm: remove ‘mtab’? y
%
```

- Μεταβλητές περιβάλλοντος (`setenv`, `unsetenv`)

```
% setenv | tail -5
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
KDEDIR=/usr
LANG=en_US
SUPPORTED=en_US:en:el_GR:el
LESSOPEN=|/usr/bin/lesspipe.sh %s
% setenv MYVAR value_of_myvar
% setenv | tail -6
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
KDEDIR=/usr
LANG=en_US
SUPPORTED=en_US:en:el_GR:el
LESSOPEN=|/usr/bin/lesspipe.sh %s
MYVAR=value_of_myvar
% echo $MYVAR
value_of_myvar
% unsetenv MYVAR
% setenv | tail -5
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
KDEDIR=/usr
LANG=en_US
SUPPORTED=en_US:en:el_GR:el
LESSOPEN=|/usr/bin/lesspipe.sh %s
% echo $MYVAR
MYVAR: Undefined variable.
%
```

- Τοπικές μεταβλητές κελύφους C (`set`, `unset`)
  `term` : Ορίζει τον τύπο του τερματικού
  `path` : Ορίζει τους καταλόγους στους οποίους το σύστημα ψάχνει για εκτελέσιμα αρχεία
  `prompt` : Ορίζει τη συμβολοσειρά που εκτυπώνει το κέλυφος C όταν περιμένει είσοδο από το χρήστη
  `noclobber` : Έλεγχος ανακατευθύνσεων σε υπάρχοντα αρχεία
  `ignoreeof` : Έλεγχος χρήσης `^D` για τερματισμό κελύφους

```
% set | grep '^[i-t]'
loginsh
noclobber
owd     /etc
path    (/usr/local/bin /bin /usr/bin /usr/X11R6/bin)
prompt  %
prompt2 %R?
prompt3 CORRECT>%R (y|n|e|a)?
shell   /bin/tcsh
shlvl   1
sourced 1
status  0
tcsh    6.10.00
term    xterm
tty     pts/0
% set term=vt100
% set path=($path /etc)
% set prompt="`hostname`/`whoami` --> "
galini/sprocsh --> set prompt="% "
% touch afile
% ls -a > afile
afile: File exists.
% unset noclobber
% ls -a > afile
% rm afile
% set ignoreeof
% ^D
Use "logout" to logout.
%
```

- Μηχανισμός ιστορίας
  - Τοπική μεταβλητή `history` χελύφους C
  - Εντολή `history`

```
% set history=5
% ls -a .*bash*
.bash_logout  .bash_profile  .bashrc
% date
Sun Feb  3 23:00:00 EET 2002
% wc .bashrc
       8      21     124 .bashrc
% echo testing...
testing...
% history
     2  22:59   ls -a .*bash*
     3  23:00   date
     4  23:00   wc .bashrc
     5  23:00   echo testing...
     6  23:00   history
% !5
echo testing...
testing...
% !da
date
Sun Feb  3 23:00:35 EET 2002
% hostname
galini
% !!
hostname
galini
% whroami
whroami: Command not found.
% ^hr^h
whoami
sprocsh
%
```

- Προσαρμογή εντολών (alias, unalias)

```
% alias dir ls
% alias rm 'rm -i'
% cp .bash_logout logout_file
% dir
logout_file
% rm logout_file
rm: remove `logout_file'? y
% alias llf 'ls -alF'
% llf
total 36
drwx------    3 sprocsh  users        4096 Feb  3 23:03 ./
drwxr-xr-x    8 root     root         4096 Feb  2 09:15 ../
-rw-r--r--    1 sprocsh  users          24 Feb  2 09:15 .bash_logout
-rw-r--r--    1 sprocsh  users         191 Feb  2 09:15 .bash_profile
-rw-r--r--    1 sprocsh  users         124 Feb  2 09:15 .bashrc
-rw-r--r--    1 sprocsh  users          30 Feb  3 13:54 .cshrc
-rw-r--r--    1 sprocsh  users         820 Feb  2 09:15 .emacs
drwxr-xr-x    3 sprocsh  users        4096 Feb  2 09:15 .kde/
-rw-r--r--    1 sprocsh  users        3511 Feb  2 09:15 .screenrc
% alias pp 'lpr -h -Plp2'
% pp .bash_profile
% alias cd 'cd \!* ; echo $cwd'
% cd /usr/lib
/usr/lib
% cd
/home/sprocsh
% alias
cd      cd !* ; echo $cwd
dir     ls
l.      ls -d .[a-zA-Z]* --color=tty
ll      ls -l --color=tty
llf     ls -alF
ls      ls --color=tty
pp      lpr -h -Plp2
rm      rm -i
% unalias rm
% unalias cd
%
```

Έλεγχος διεργασιών και εργασιών

`ps` : Εμφάνιση κατάστασης τρεχουσών διεργασιών
`jobs` : Εμφάνιση ενεργών εργασιών
`kill` : Τερματισμός διεργασιών και εργασιών
`bg` : Μεταφορά εργασιών στο παρασκήνιο
`fg` : Μεταφορά εργασιών στο προσκήνιο

Εντολές ps (επιλογή -u),
jobs,
kill (επιλογή -9)

```
$ ps
  PID TTY          TIME CMD
 1052 pts/0    00:00:00 bash
 1093 pts/0    00:00:00 ps
$ ps -u
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
spro      1052  0.7  2.1  2488 1344 pts/0    S    21:08   0:00 -bash
spro      1094  0.0  1.1  2600  716 pts/0    R    21:09   0:00 ps -u
$ cp -r /usr/share/icons .
^Z
[1]+  Stopped                 cp -r /usr/share/icons .
$ ps -u
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
spro      1052  0.4  2.1  2488 1348 pts/0    S    21:08   0:00 -bash
spro      1095  1.8  0.9  1696  604 pts/0    T    21:09   0:00 cp -r
                                                           /usr/share/
spro      1096  0.0  1.1  2600  716 pts/0    R    21:09   0:00 ps -u
$ jobs
[1]+  Stopped                 cp -r /usr/share/icons .
$ kill %1

[1]+  Stopped                 cp -r /usr/share/icons .
$
[1]+  Terminated              cp -r /usr/share/icons .
$ cp -r /usr/lib . &
[1] 1097
$ ps
  PID TTY          TIME CMD
 1052 pts/0    00:00:00 bash
 1097 pts/0    00:00:00 cp
 1098 pts/0    00:00:00 ps
$ kill -9 1097
$
[1]+  Killed                  cp -r /usr/lib .
$ ps
  PID TTY          TIME CMD
 1052 pts/0    00:00:00 bash
 1099 pts/0    00:00:00 ps
$ rm -r icons lib
$
```

Εντολές bg, fg

```
$ cp -r /usr/share .
^Z
[1]+  Stopped                 cp -r /usr/share .
$ bg
[1]+ cp -r /usr/share . &
$ jobs
[1]+  Running                 cp -r /usr/share . &
$ ps
  PID TTY          TIME CMD
 1052 pts/0    00:00:00 bash
 1102 pts/0    00:00:01 cp
 1103 pts/0    00:00:00 ps
$ fg
cp -r /usr/share .
^Z
[1]+  Stopped                 cp -r /usr/share .
$ bg
[1]+ cp -r /usr/share . &
$ ps
  PID TTY          TIME CMD
 1052 pts/0    00:00:00 bash
 1102 pts/0    00:00:01 cp
 1104 pts/0    00:00:00 ps
$ kill 1102
$
[1]+  Terminated              cp -r /usr/share .
$ ps
  PID TTY          TIME CMD
 1052 pts/0    00:00:00 bash
 1105 pts/0    00:00:00 ps
$ chmod -R 755 share
$ rm -r share
$
```

Εφεδρικά, συμπιεσμένα, κωδικοποιημένα αρχεία

`tar`	: Χειρισμός εφεδρικών αρχείων
`gzip`	: Συμπίεση αρχείων
`gunzip`	: Αποσυμπίεση αρχείων
`zcat`	: Εμφάνιση περιεχομένων συμπιεσμένων αρχείων
`uuencode`	: Κωδικοποίηση αρχείων
`uudecode`	: Αποκωδικοποίηση αρχείων

Εντολή tar (επιλογές f, c, t, x, v)

```
$ pushd /
/ ~
$ tar cvf ~/file.tar var/yp
var/yp/
var/yp/binding/
var/yp/binding/di.uoa.gr.1
var/yp/binding/di.uoa.gr.2
var/yp/nicknames
$ popd
~
$ tar tf file.tar
var/yp/
var/yp/binding/
var/yp/binding/di.uoa.gr.1
var/yp/binding/di.uoa.gr.2
var/yp/nicknames
$ tar xvf file.tar
var/yp/
var/yp/binding/
var/yp/binding/di.uoa.gr.1
var/yp/binding/di.uoa.gr.2
var/yp/nicknames
$ ls -R
.:
bin  file.tar  var

./bin:

./var:
yp

./var/yp:
binding  nicknames

./var/yp/binding:
di.uoa.gr.1  di.uoa.gr.2
$ rm -r file.tar var
$
```

Εντολές gzip (επιλογή -v), gunzip, zcat

```
$ cp /etc/termcap .
$ ls -l
total 732
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users      737535 Feb  5 21:29 termcap
$ gzip termcap
$ ls -l
total 244
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users      238036 Feb  5 21:29 termcap.gz
$ zcat termcap.gz | head -700 | tail -8
        :ue=\E[24m:up=\E[A:us=\E[4m:vb=200\E[?5h\E[?5l:\
        :ve=\E[?25h\E[?0c:vi=\E[?25l\E[?1c:vs=\E[?25h\E[?8c:\
        :tc=klone+sgr:tc=ecma+color:
linux-m|Linux console no color:\
        :Co@:pa@:\
        :AB@:AF@:Sb@:Sf@:tc=linux:
linux-c-nc|linux console 1.3.x hack for ncurses only:\
        :cc:\
$ gunzip *.gz
$ ls -l
total 732
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rw-r--r--    1 spro     users      737535 Feb  5 21:29 termcap
$ gzip -v termcap
termcap:                 67.7% -- replaced with termcap.gz
$ rm termcap.gz
$
```

Εντολές uuencode, uudecode

```
$ cp /bin/tcsh .
$ uuencode tcsh tcsh.dupl > tcsh.uue
$ ls -l
total 688
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
-rwxr-xr-x    1 spro     users      288604 Feb  5 21:33 tcsh
-rw-r--r--    1 spro     users      397662 Feb  5 21:33 tcsh.uue
$ head -3 tcsh.uue
begin 755 tcsh.dupl
M?T5,1@$!‘0‘‘‘‘‘‘‘‘‘‘‘(‘‘P‘!‘‘‘‘\*$$"#0‘‘‘!T8P0‘‘‘‘‘‘#0‘(‘‘&
M‘"@‘&0‘8‘‘8‘‘‘‘T‘‘‘‘-(‘$"#2‘!‘C‘‘‘‘‘P‘‘‘‘‘4‘‘‘‘$‘‘‘‘‘P‘‘‘/0‘
$ tail -4 tcsh.uue
MH&($‘"!!‘P‘‘‘‘‘‘‘‘‘‘"‘‘‘‘‘‘‘‘‘‘$0‘‘‘‘,‘‘‘‘‘‘‘‘‘‘‘‘‘*!B!‘#4
3‘‘‘‘‘‘‘‘‘‘‘‘‘‘!‘‘‘‘‘‘‘‘‘‘‘
‘
end
$ uudecode tcsh.uue
$ ls
bin  tcsh  tcsh.dupl  tcsh.uue
$ cmp tcsh tcsh.dupl
$ rm tcsh*
$ pushd /usr/share
/usr/share ~
$ tar cf ~/man.tar man
$ popd
~
$ ls -l man*
-rw-r--r--    1 spro     users     9881600 Feb  5 21:55 man.tar
$ gzip man.tar
$ ls -l man*
-rw-r--r--    1 spro     users     7766170 Feb  5 21:55 man.tar.gz
$ uuencode man.tar.gz mandpl.tar.gz > man.tar.gz.uue
$ ls -l man*
-rw-r--r--    1 spro     users     7766170 Feb  5 21:55 man.tar.gz
-rw-r--r--    1 spro     users    10700090 Feb  5 21:57 man.tar.gz.uue
$ uudecode man.tar.gz.uue
$ ls -l man*
-rw-r--r--    1 spro     users     7766170 Feb  5 21:57 mandpl.tar.gz
-rw-r--r--    1 spro     users     7766170 Feb  5 21:55 man.tar.gz
-rw-r--r--    1 spro     users    10700090 Feb  5 21:57 man.tar.gz.uue
$ gunzip mandpl.tar.gz
$ tar tvf mandpl.tar | grep du.1.gz
-rw-r--r-- root/root      1234 2001-08-09 16:01:17 man/man1/du.1.gz
-rw-r--r-- root/root      1112 2001-06-24 23:11:51 man/man1/mdu.1.gz
$ rm man*
$
```

Μερικές ακόμα εντολές του Unix

`du` : Εμφάνιση χρήσης του δίσκου

`find` : Ανεύρεση αρχείων ή καταλόγων

`file` : Προσδιορισμός είδους αρχείων

`od` : Εμφάνιση περιεχομένων αρχείων με διάφορους τρόπους

`tr` : Μετατροπή χαρακτήρων σε αρχεία

`colrm` : Διαγραφή στηλών από αρχεία

`last` : Εμφάνιση πιο πρόσφατων συνδέσεων από χρήστες

`script` : Καταγραφή της τρέχουσας αλληλεπίδρασης με το κέλυφος

`sleep` : Αναμονή για προκαθορισμένο χρόνο

`clear` : Καθάρισμα της οθόνης

Εντολές du (επιλογές -s, -a),
find (επιλογές -name, -exec, -print),
file

```
$ du /boot
72      /boot/grub
1384    /boot
$ du -s .
40      .
$ du -a /etc/midi
112     /etc/midi/GU11-ROM.SF2
8       /etc/midi/drums.o3
8       /etc/midi/drums.sb
8       /etc/midi/std.o3
8       /etc/midi/std.sb
148     /etc/midi
$ find /usr -name passwd -print
/usr/bin/passwd
/usr/share/doc/nss_ldap-172/pam.d/passwd
/usr/share/doc/pam_krb5-1.46/krb5afs-pam.d/passwd
/usr/share/doc/pam_krb5-1.46/pam.d/passwd
$ find /usr/share -name '*jk[a-z]*' -print -exec cp {} . \;
/usr/share/doc/4Suite-0.11/docs/text/howto/cjkv
/usr/share/doc/4Suite-0.11/docs/xml/howto/cjkv.doc
/usr/share/emacs/20.7/lisp/jka-compr.elc
$ ls
bin  cjkv  cjkv.doc  jka-compr.elc
$ find ~ -name '*k*' -print -exec rm {} \;
/home/spro/cjkv
/home/spro/cjkv.doc
/home/spro/jka-compr.elc
$ ls
bin
$ pushd /usr/share/ssl
/usr/share/ssl ~
$ file *s*
certs:       directory
misc:        directory
openssl.cnf: ASCII English text
$ find . -name '*s*' -exec file {} \;
./misc: directory
./misc/c_hash: Bourne shell script text executable
./misc/c_issuer: Bourne shell script text executable
./certs: directory
./openssl.cnf: ASCII English text
$ popd
~
$
```

Εντολή od (επιλογές -d, -h, -c)

```
$ cat /etc/filesystems
ext3
ext2
nodev proc
nodev devpts
iso9660
vfat
hfs
$ od /etc/filesystems
0000000 074145 031564 062412 072170 005062 067556 062544 020166
0000020 071160 061557 067012 062157 073145 062040 073145 072160
0000040 005163 071551 034557 033066 005060 063166 072141 064012
0000060 071546 000012
0000063
$ od -d /etc/filesystems
0000000 30821 13172 25866 29816  2610 28526 25956  8310
0000020 29296 25455 28170 25711 30309 25632 30309 29808
0000040  2675 29545 14703 13878  2608 26230 29793 26634
0000060 29542    10
0000063
$ od -h /etc/filesystems
0000000 7865 3374 650a 7478 0a32 6f6e 6564 2076
0000020 7270 636f 6e0a 646f 7665 6420 7665 7470
0000040 0a73 7369 396f 3636 0a30 6676 7461 680a
0000060 7366 000a
0000063
$ od -c /etc/filesystems
0000000   e  x  t  3 \n  e  x  t  2 \n  n  o  d  e  v
0000020   p  r  o  c \n  n  o  d  e  v     d  e  v  p  t
0000040   s \n  i  s  o  9  6  6  0 \n  v  f  a  t \n  h
0000060   f  s \n
0000063
$ od -hc /etc/filesystems
0000000 7865 3374 650a 7478 0a32 6f6e 6564 2076
          e  x  t  3 \n  e  x  t  2 \n  n  o  d  e  v
0000020 7270 636f 6e0a 646f 7665 6420 7665 7470
          p  r  o  c \n  n  o  d  e  v     d  e  v  p  t
0000040 0a73 7369 396f 3636 0a30 6676 7461 680a
          s \n  i  s  o  9  6  6  0 \n  v  f  a  t \n  h
0000060 7366 000a
          f  s \n \0
0000063
$
```

Εντολές tr (επιλογή -d), colrm

```
$ cat /etc/resolv.conf
nameserver 195.134.65.119
search di.uoa.gr
domain di.uoa.gr
$ tr 56789abc 012345xyz < /etc/resolv.conf
n5meserver 140.134.10.114
se5ryh di.uo5.gr
dom5in di.uo5.gr
$ tr abcdefghijklm +-@ < /etc/resolv.conf
n+7/s/rv/r 195.134.65.119
s/+r-2 .3.uo+.1r
.o7+3n .3.uo+.1r
$ tr -d '[0-9]uoa' < /etc/resolv.conf
nmeserver ...
serch di..gr
dmin di..gr
$ colrm 8 14 < /etc/resolv.conf
nameser.134.65.119
search gr
domain gr
$ colrm 15 < /etc/resolv.conf
nameserver 195
search di.uoa.
domain di.uoa.
$ ls -ald bin
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
$ ls -ald bin | colrm 11 55
drwxr-xr-x bin
$
```

Εντολές last (επιλογή -n), script, sleep, clear

```
$ last -n 8
spro     pts/0       ppp-116.dialup.u Wed Feb  6 21:28 still logged in
reboot   system boot 2.4.7-10         Wed Feb  6 16:19        (05:33)
spro     pts/0       knossos.di.uoa.g Wed Feb  6 12:24-12:27  (00:03)
spro     pts/0       ppp-117.dialup.u Wed Feb  6 10:48-11:03  (00:14)
spro     pts/0       ppp-100.dialup.u Wed Feb  6 09:21-10:20  (00:59)
spro     pts/0       ppp-50.dialup.uo Tue Feb  5 21:08-22:00  (00:52)
reboot   system boot 2.4.7-10         Tue Feb  5 16:29        (22:19)
sprocsh  pts/1       ppp-146.dialup.u Mon Feb  4 22:33-22:49  (00:16)

wtmp begins Fri Feb  1 12:27:24 2002
$ script my_session
Script started, file is my_session
$ date
Wed Feb  6 21:56:29 EET 2002
$ whoami
spro
$ exit
exit
Script done, file is my_session
$ cat my_session
Script started on Wed Feb  6 21:56:25 2002
$ date
Wed Feb  6 21:56:29 EET 2002
$ whoami
spro
$ exit
exit

Script done on Wed Feb  6 21:56:37 2002
$ rm my_session
$ date ; sleep 20 ; date
Wed Feb  6 21:57:24 EET 2002
Wed Feb  6 21:57:44 EET 2002
$ clear
```

Χρήσιμα προγράμματα του Unix

`mail` : Ηλεκτρονικό ταχυδρομείο
`cc` : Μεταγλωττιστής της C
.....
`sed` : Μαζικός διορθωτής [§4.3–**KP**]
`awk` : Επεξεργαστής συμβολοσειρών [§4.4–**KP**]

Ηλεκτρονικό ταχυδρομείο `mail` (επιλογή `-s`)

```
% mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/syspro": 1 message 1 new
>N  1 takis@di.uoa.gr       Wed Feb  6 22:24  24/1000  "Testing"
& 1
Message 1:
From takis@di.uoa.gr Wed Feb  6 22:24:56 2002
From: Takis Stamatopoulos <takis@di.uoa.gr>
Subject: Testing
To: syspro@di.uoa.gr
Date: Wed, 6 Feb 2002 22:24:44 +0200 (EET)
X-Mailer: ELM [version 2.4ME+ PL66 (25)]
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
X-MailScanner: Found to be clean

This is a test message to demonstrate the
use of 'mail'.

& s 1 mess
"mess" [New file]
& d 1
& q
% mail takis@di.uoa.gr
Subject: Thanks
OK, I received it.
Cc:
% mail -s 'My login' takis@di.uoa.gr < .login
% ls -l mess
-rw-------    1 syspro   users        1010 Feb  6  2002 mess
% rm mess
% mail
No mail for syspro
%
```

Μεταγλωττιστής της C cc ή gcc (επιλογές -o, -c)

```
$ ls
bin  main.c  syntax.c  syntax.h  tokens.c  tokens.h  world.c
$ cat world.c
# include <stdio.h>

main()
{
  printf("Hello world!\n");
}
$ cc -o world world.c
$ ls
bin  main.c  syntax.c  syntax.h  tokens.c  tokens.h  world  world.c
$ ./world
Hello world!
$ cc -c main.c
$ cc -c syntax.c
$ cc -c tokens.c
$ ls
bin     main.o    syntax.h  tokens.c  tokens.o  world.c
main.c  syntax.c  syntax.o  tokens.h  world
$ cc -o parser main.o syntax.o tokens.o
$
```

Προγραμματισμός κελύφους Bourne

- Εκτελέσιμα προγράμματα/σενάρια
- Ακολουθία από εντολές
- Το # δηλώνει την αρχή σχολίου που εκτείνεται μέχρι το τέλος της γραμμής
- Πρώτη γραμμή: `#! /bin/sh`
- Συμβολοσειρές που περικλείονται μεταξύ " ή ' μπορεί να περιέχουν ειδικούς χαρακτήρες για το κέλυφος, π.χ. κενά, μεταχαρακτήρες κ.λ.π.
- Μία εντολή μπορεί να επεκταθεί σε επόμενη γραμμή, εφ' όσον διακοπεί με \
- Δυνατότητα ανακατευθύνσεων με εν μέρει διαφορετικό τρόπο απ' ότι στο κέλυφος C [§3.7–**KP**]
- Ορίσματα προγράμματος
- Μεταβλητές
- Συνθήκες
- Δομές ελέγχου
- Υπολογισμός εκφράσεων

Ορίσματα προγράμματος

- Η έκφραση `$0` αντιπροσωπεύει το όνομα του προγράμματος
- Οι εκφράσεις `$1`, `$2`, ... , `$9` αντιπροσωπεύουν το 1ο, 2ο, ... , 9ο όρισμα που δόθηκε κατά την εκτέλεση του προγράμματος
- Η εντολή `shift` μπορεί να χρησιμοποιηθεί σε περίπτωση περισσότερων από 9 ορίσματα
- Η έκφραση `$#` αντιπροσωπεύει το πλήθος των ορισμάτων που δόθηκαν κατά την εκτέλεση του προγράμματος
- Η έκφραση `$*` αντιπροσωπεύει όλα τα ορίσματα που δόθηκαν, χωρισμένα μεταξύ τους με έναν κενό χαρακτήρα

Μεταβλητές

- Η εντολή ⟨μεταβλητή⟩=⟨τιμή⟩ αναθέτει την ⟨τιμή⟩ στη ⟨μεταβλητή⟩
- Η έκφραση $⟨μεταβλητή⟩ αντιπροσωπεύει την τιμή που έχει η ⟨μεταβλητή⟩
- Η εντολή `read` ⟨μεταβλητή⟩ αναθέτει την τιμή που δίνεται για διάβασμα από την προκαθορισμένη είσοδο στη ⟨μεταβλητή⟩
- Υπάρχουν προκαθορισμένες μεταβλητές περιβάλλοντος, όπως οι `PATH`, `HOME` κ.λ.π.
- Η έκφραση ‘⟨εντολή⟩‘ αντιπροσωπεύει το αποτέλεσμα που έχει η ⟨εντολή⟩ όταν εκτελεσθεί, σαν μία λίστα από λέξεις χωρισμένες με κενούς χαρακτήρες
- Όταν μία ⟨μεταβλητή⟩ έχει σαν τιμή μία λίστα από λέξεις χωρισμένες με κενούς χαρακτήρες, η εντολή `set -` $⟨μεταβλητή⟩ αναθέτει τις λέξεις αυτές σαν τιμές στις μεταβλητές `$1`, `$2`, ... κατά σειρά

Συνθήκες

- Σύνταξη: `[` ⟨έλεγχος⟩ `]` ή
  `test` ⟨έλεγχος⟩
- Είδη
  - Σε αρχεία (`-r`, `-w`, `-x`, `-f`, `-d`, `-s`)
    π.χ. `test -x my_file`
  - Σε συμβολοσειρές (`-z`, `-n`, `=`, `!=`)
    π.χ. `[ "$var" != stri ]`
  - Σε ακεραίους (`-eq`, `-ne`, `-gt`, `-lt`, `-le`, `-ge`)
    π.χ. `test $1 -ge 0`
  - Σύνθετες (`!`, `-a`, `-o`)
    π.χ. `[ "$bla" = f -a $foo -eq 2 ]`
- Ομαδοποίηση με `\(` και `\)`

Δομές ελέγχου

- Δομή `if`
  - Σύνταξη: `if` $\langle$συνθήκη$\rangle_1$
    `then`
    $\langle$εντολές$\rangle_1$
    `elif` $\langle$συνθήκη$\rangle_2$
    `then`
    $\langle$εντολές$\rangle_2$
    $\vdots$
    `elif` $\langle$συνθήκη$\rangle_{\nu\text{-}1}$
    `then`
    $\langle$εντολές$\rangle_{\nu\text{-}1}$
    `else`
    $\langle$εντολές$\rangle_\nu$
    `fi`
  - Τα τμήματα `elif` και/ή `else` μπορεί και να μην υπάρχουν

- Δομή `case`
  - Σύνταξη: `case` ⟨έκφραση⟩ `in`
    ⟨περίπτωση⟩$_1$`)` ⟨εντολές⟩$_1$ `;;`
    ⟨περίπτωση⟩$_2$`)` ⟨εντολές⟩$_2$ `;;`
    ⋮
    ⟨περίπτωση⟩$_\nu$`)` ⟨εντολές⟩$_\nu$ `;;`
    `esac`
  - Κάθε ⟨περίπτωση⟩$_\iota$ είναι πρότυπο για ταίριασμα το οποίο μπορεί να περιλαμβάνει και μεταχαρακτήρες
  - Το `|` δηλώνει διάζευξη

- Δομές ανακυκλώσεων
  - Σύνταξη: ⟨δομή ανακυκλώσεων⟩
    ```
    do
      ⟨εντολές⟩
    done
    ```
  - Δυνατότητες για τη ⟨δομή ανακυκλώσεων⟩
    * `for` ⟨μεταβλητή⟩
    * `for` ⟨μεταβλητή⟩ `in` ⟨λίστα⟩
    * `while` ⟨συνθήκη⟩
    * `until` ⟨συνθήκη⟩
  - Η εντολή `break` διακόπτει τις ανακυκλώσεις
  - Η εντολή `continue` μεταφέρει τον έλεγχο στο τέλος της τρέχουσας ανακύκλωσης
- Η εντολή `exit` τερματίζει την εκτέλεση ενός προγράμματος

Υπολογισμός εκφράσεων

- Βοηθητικό πρόγραμμα `expr`
- Αριθμητικοί τελεστές `+`, `-`, `*`, `/`, `%`
  - Παραδείγματα
    - `expr 8 - 3 \* 2` 2
    - `expr \( 8 - 3 \) \* 2` 10
    - `expr 10 / 3 + 4` 7
    - `expr 20 % \( 7 + 2 \)` 2
- Τελεστές συμβολοσειρών `substr`, `index`, `length`
  - Παραδείγματα
    - `expr substr transputer 4 5` `nsput`
    - `expr index smalltalk btr` 6
    - `expr length systems` 7

Διάφορα προβλήματα

- *Να γραφεί ένα πρόγραμμα για το κέλυφος Bourne που να εμφανίζει τους καταλόγους που βρίσκονται κάτω από δεδομένο κατάλογο, σε οποιοδήποτε βάθος, καθώς επίσης και την ημερομηνία και ώρα τελευταίας τροποποίησής τους (ή δημιουργίας τους).*

```
#! /bin/sh
#
# Usage: lsdir directory
#
if [ $# -eq 0 ]
then
   echo "Exactly one argument is required"
elif [ $# -ge 2 ]
then
   echo "Too many arguments"
else
   ls -Rl $1 | grep '^d' | colrm 1 43
fi
```

- *Να γραφεί ένα πρόγραμμα για το κέλυφος Bourne που να εκτελεί απλές αριθμητικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση και υπόλοιπο) μεταξύ δύο ακεραίων.*

```
#! /bin/sh
#
# Usage: math n1 op n2
#
case $2 in
   +)  echo "Addition requested."
       echo "$1 + $3 = `expr $1 + $3`" ;;
   -)  echo "Substraction requested."
       echo "$1 - $3 = `expr $1 - $3`" ;;
  \*)  echo "Multiplication requested."
       echo "$1 * $3 = `expr $1 \* $3`" ;;
   /)  echo "Division requested."
       echo "$1 / $3 = `expr $1 / $3`" ;;
   %)  echo "Modulo arithmetic requested."
       echo "$1 % $3 = `expr $1 % $3`" ;;
   *)  echo "Unknown operation specified." ;;
esac
```

- *Να γραφεί ένα πρόγραμμα για το κέλυφος Bourne που να εκτυπώνει στην οθόνη κάποιο μήνυμα μόνο την πρώτη φορά που εκτελείται κατά τη διάρκεια μίας ημέρας.*

```
#! /bin/sh
#
# Usage: once
#
cur_date=‘date +%d%m%y‘
last_date=""
if [ -s $HOME/.date ]
then
   last_date=‘cat $HOME/.date‘
fi
echo "$cur_date" > $HOME/.date
if [ "$last_date" != "$cur_date" ]
then
   echo "Hello!! How are you today?"
fi
```

- *Να γραφεί ένα πρόγραμμα για το κέλυφος Bourne που να υπολογίζει το παραγοντικό ενός ακεραίου αριθμού.*

```
#! /bin/sh
#
# Usage: factorial
#
echo -n "Give input number: "
read n
fact=1
until [ $n -eq 0 ]
do
   fact=‘expr $fact \* $n‘
   n=‘expr $n - 1‘
done
echo $fact
```

- *Να γραφεί ένα πρόγραμμα για το κέλυφος Bourne που να αντιστρέφει δεδομένες συμβολοσειρές, καθώς επίσης να υπολογίζει και τα μήκη τους.*

```
#! /bin/sh
#
# Usage: revstrs [string1 [string2 ...]]
#
for str
do
   strlen=‘expr length "$str"‘
   chind=$strlen
   while test $chind -gt 0
   do
      echo -n "‘expr substr \"$str\" $chind 1‘"
      chind=‘expr $chind - 1‘
   done
   echo -n " --> "
   echo -n "$strlen"
   echo " character(s)."
done
```

- *Να γραφεί ένα πρόγραμμα για το κέλυφος Bourne που να υπολογίζει το μέγιστο χώρο που καταλαμβάνουν τα περιεχόμενα δεδομένων καταλόγων.*

```
#! /bin/sh
#
# Usage: maxsize
#
echo -n "Please specify the directory names: "
read dirs ; set - $dirs ; ndirs=$#
max=0
while [ $ndirs -ne 0 ]
do
   dir=$1 ; shift ; dirs=$*
   if [ ! -d $dir ]
   then
      echo "$0 : There is no directory $dir"
   else
      duout=‘du -s $dir‘ ; set - $duout ; size=$1
      if [ $size -gt $max ]
      then
         max=$size ; maxdir=$dir
      fi
   fi
   set - $dirs
   ndirs=‘expr $ndirs - 1‘
done
echo "$maxdir $max"
```

Προγραμματισμός λειτουργιών του Unix σε C

- Συναρτήσεις βιβλιοθήκης
- Κλήσεις συστήματος
- Κατηγορίες λειτουργιών
  - Χειρισμός λαθών
  - Δημιουργία και τερματισμός διεργασιών
  - Αποστολή/παραλαβή σημάτων
  - Είσοδος/έξοδος χαμηλού επιπέδου
  - Επικοινωνία μεταξύ διεργασιών μέσω
    * σωλήνων
    * υποδοχών
    * ουρών μηνυμάτων
    * κοινής μνήμης
    * σηματοφόρων
  - Δημιουργία, τερματισμός και συγχρονισμός νημάτων
  - Διαχείριση συστήματος αρχείων

Χειρισμός λαθών

- Νόμος του Murphy
  - *If anything can go wrong, it will*
- Ο σωστός προγραμματισμός απαιτεί έλεγχο των περιπτώσεων λάθους που μπορεί να συμβούν και ανάλογη δράση
- Συνάρτηση βιβλιοθήκης `perror`
  - `void perror(char *str)`
  - Εκτυπώνει τη συμβολοσειρά `str` και μία περιγραφή του πιο πρόσφατου λάθους που έχει συμβεί
- Εξωτερική μεταβλητή `errno`
  - Εχει σαν τιμή έναν ακέραιο που αντιστοιχεί στο πιο πρόσφατο λάθος που έχει συμβεί
  - Απαίτηση: `#include <errno.h>`

- Χρήση της συνάρτησης perror και της εξωτερικής μεταβλητής errno

```
/* File: errors_demo.c */
#include <stdio.h>              /* For fopen, printf */
#include <stdlib.h>                    /* For malloc */
#include <errno.h>             /* For errno variable */
main()
{  FILE *fp; char *p; int stat;
   fp = fopen("non_existent_file", "r");
   if (fp == NULL) {              /* Check for error */
      printf("errno = %d\n", errno);
      perror("fopen"); }
   p = malloc(4000000000U);
   if (p == NULL) {               /* Check for error */
      printf("errno = %d\n", errno);
      perror("malloc"); }
   /***** BE CAREFUL: unlink tries to remove a file */
   stat = unlink("/etc/motd");
   if (stat == -1) {              /* Check for error */
      printf("errno = %d\n", errno);
      perror("unlink"); } }
```

```
$ ./errors_demo
errno = 2
fopen: No such file or directory
errno = 12
malloc: Cannot allocate memory
errno = 13
unlink: Permission denied
$
```

Διαχείριση διεργασιών

- Κάθε διεργασία στο Unix έχει έναν αριθμό ταυτότητας (PID), τον κώδικά της, τα δεδομένα της, μία στοίβα καθώς και κάποια άλλα χαρακτηριστικά
- Η αρχική διεργασία είναι η init (PID=1)
- Ο μόνος τρόπος να δημιουργηθεί μία διεργασία είναι κάποια άλλη να αναπαραγάγει τον εαυτό της, δηλαδή μία διεργασία-γονέας να γεννήσει μία διεργασία-παιδί
- Όλες οι διεργασίες είναι απόγονοι της init
- Ο κώδικας, τα δεδομένα και η στοίβα της διεργασίας-παιδιού είναι αντίγραφα αυτών της διεργασίας-γονέα
- Η ταυτότητα της διεργασίας-παιδιού είναι διαφορετική από την ταυτότητα της διεργασίας-γονέα
- Μία διεργασία-παιδί μπορεί να αντικαταστήσει τον κώδικα, τα δεδομένα και τη στοίβα της με αυτά ενός εκτελέσιμου αρχείου, διαφοροποιώντας έτσι τον εαυτό της από το γονέα της

- Κλήση συστήματος `exit`
  - `void exit(int status)`
  - Τερματίζει την εκτέλεση μίας διεργασίας
  - Ο ακέραιος `status` ονομάζεται κωδικός εξόδου και είναι διαθέσιμος στη διεργασία-γονέα
  - Κατά σύμβαση, η τιμή 0 για το `status` δείχνει επιτυχή τερματισμό
  - Πρόσβαση με το `$status` στο κέλυφος C και με το `$?` στο κέλυφος Bourne
  - Απαίτηση: `#include <stdlib.h>`
- Χρήση της κλήσης `exit`

```
/* File: exit_code.c */
#include <stdio.h>   /* For printf */
#include <stdlib.h>    /* For exit */
main()
{  printf("I am going to terminate with exit code 23\n");
   exit(23); }
```

```
$ ./exit_code
I am going to terminate with exit code 23
$ echo $?
23
$
```

- Κλήση συστήματος `fork`
  - `int fork()`
  - Αναπαράγει μία διεργασία, δημιουργώντας μία άλλη πανομοιότυπη
  - Στη διεργασία-γονέα επιστρέφει την ταυτότητα της διεργασίας-παιδιού και στη διεργασία-παιδί επιστρέφει 0
  - Επιστρέφει λάθος (-1) στη διεργασία-γονέα αν δεν είναι δυνατόν να δημιουργηθεί η νέα διεργασία-παιδί
- Κλήσεις συστήματος `getpid` και `getppid`
  - `int getpid()`
  - `int getppid()`
  - Επιστρέφουν τις ταυτότητες μίας διεργασίας και του γονέα της αντίστοιχα
- Οι κλήσεις συστήματος `fork`, `getpid` και `getppid` (αλλά και η `wait` που θα ακολουθήσει) τυπικά επιστρέφουν `pid_t` (χρειάζεται και `#include <sys/types.h>`), αλλά το `pid_t` είναι `typedef`'ed σε `int` στα περισσότερα συστήματα

- Χρήση των κλήσεων fork, getpid και getppid

```
/* File: fork_a_process.c */
#include <stdio.h>                      /* For printf */
#include <stdlib.h>                       /* For exit */
main()
{  int pid;
   printf("Original process: PID = %d, PPID = %d\n",
          getpid(), getppid());
   pid = fork();
   if (pid == -1) {               /* Check for error */
      perror("fork");
      exit(1); }
   if (pid != 0) {             /* The parent process */
      printf("Parent process: PID = %d, PPID = %d, CPID = %d\n",
             getpid(), getppid(), pid); }
   else {                        /* The child process */
      printf("Child process: PID = %d, PPID = %d\n",
             getpid(), getppid()); }
   printf("Process with PID = %d terminates\n",
          getpid()); } /* Executed by both processes */
```

```
$ ./fork_a_process
Original process: PID = 1585, PPID = 1541
Parent process: PID = 1585, PPID = 1541, CPID = 1586
Child process: PID = 1586, PPID = 1585
Process with PID = 1586 terminates
Process with PID = 1585 terminates
$
```

- Αν μία διεργασία τερματίσει πρίν από κάποιο παιδί της, τότε το τελευταίο, σαν ορφανή (orphan) διεργασία, υιοθετείται από την init

```
/* File: orphan_process.c */
#include <stdio.h>                              /* For printf */
#include <stdlib.h>                               /* For exit */
main()
{  int pid;
   printf("Original process: PID = %d, PPID = %d\n",
          getpid(), getppid());
   pid = fork();
   if (pid == -1) {                      /* Check for error */
      perror("fork");
      exit(1); }
   if (pid != 0) {                     /* The parent process */
      printf("Parent process: PID = %d, PPID = %d, CPID = %d\n",
             getpid(), getppid(), pid); }
   else {                                /* The child process */
      sleep(2);          /* Delay child in order to ensure
                             that the parent terminates first */
      printf("Child process: PID = %d, PPID = %d\n",
             getpid(), getppid()); }
   printf("Process with PID = %d terminates\n",
          getpid()); }       /* Executed by both processes */
```

```
$ ./orphan_process
Original process: PID = 1587, PPID = 1541
Parent process: PID = 1587, PPID = 1541, CPID = 1588
Process with PID = 1587 terminates
$ Child process: PID = 1588, PPID = 1
Process with PID = 1588 terminates
```

- Κλήση συστήματος `wait`
  - `int wait(int *status)`
  - Προκαλεί την αναμονή μίας διεργασίας μέχρις ότου κάποιο παιδί της τερματίσει
  - Αποδέχεται τον κωδικό εξόδου του παιδιού, δηλαδή τον ακέραιο της κλήσης `exit`, στο αριστερό byte του `status`, ενώ το δεξιό byte είναι 0
  - Αν το παιδί τερματίσει εξ αιτίας κάποιου σήματος, τότε τα 7 τελευταία bits του `status` αντιπροσωπεύουν το χαρακτηριστικό αριθμό αυτού του σήματος
  - Επιστρέφει την ταυτότητα του παιδιού που τερμάτισε ή λάθος (-1) αν η διεργασία δεν έχει παιδιά

- Χρήση της κλήσης `wait`

```
/* File: wait_usage.c */
#include <stdio.h>                              /* For printf */
#include <stdlib.h>                               /* For exit */
main()
{  int pid, status;
   printf("Original process: PID = %d\n", getpid());
   pid = fork();
   if (pid == -1) {                        /* Check for error */
      perror("fork");
      exit(1); }
   if (pid != 0) {                       /* The parent process */
      printf("Parent process: PID = %d\n", getpid());
      if (wait(&status) != pid) {  /* Wait for child to exit */
         perror("wait");
         exit(1); }
      printf("Child terminated: PID = %d, exit code = %d\n",
             pid, status >> 8); }
   else {                                  /* The child process */
      printf("Child process: PID = %d, PPID = %d\n",
             getpid(), getppid());
      exit(72); }                /* Exit with a silly number */
   printf("Process with PID = %d terminates\n",
          getpid()); }            /* Executed by parent only */
```

```
$ ./wait_usage
Original process: PID = 1591
Child process: PID = 1592, PPID = 1591
Parent process: PID = 1591
Child terminated: PID = 1592, exit code = 72
Process with PID = 1591 terminates
$
```

- Μία διεργασία που τερματίζει δεν εγκαταλείπει το σύστημα μέχρις ότου ο γονέας της αποδεχθεί τον κωδικό εξόδου της και είναι όλο αυτό το χρονικό διάστημα μία ζωντανή-νεκρή (zombie) διεργασία

```
/* File: zombie_process.c */
#include <stdlib.h>                    /* For exit */
main()
{  int pid;
   pid = fork();
   if (pid == -1) {              /* Check for error */
      perror("fork");
      exit(1); }
   if (pid != 0) {           /* The parent process */
      while (1)                  /* Never terminate */
         sleep(1000); }
   else {                      /* The child process */
      exit(37); } }    /* Exit with a silly number */
```

```
$ ./zombie_process &
[1] 1593
$ ps -a
  PID TTY          TIME CMD
 1593 pts/0    00:00:00 zombie_process
 1594 pts/0    00:00:00 zombie_process <defunct>
 1597 pts/0    00:00:00 ps
$ kill 1593
$
[1]+  Terminated              ./zombie_process
$ ps -a
  PID TTY          TIME CMD
 1600 pts/0    00:00:00 ps
$
```

- Κλήσεις συστήματος της ομάδας `exec`
  - `int execl(char *path, char *arg0, char *arg1, ... , char *argn, NULL)`
  - `int execv(char *path, char *argv[])`
  - `int execlp(char *path, char *arg0, char *arg1, ... , char *argn, NULL)`
  - `int execvp(char *path, char *argv[])`
  - Αντικαθιστούν μία διεργασία (κώδικα, δεδομένα, στοίβα) με αυτά του εκτελέσιμου αρχείου που δίνεται στο `path`
  - Οι `execl` και `execv` απαιτούν απόλυτα ή σχετικά ονόματα-μονοπάτια στο `path`
  - Οι `execlp` και `execvp` χρησιμοποιούν τη μεταβλητή περιβάλλοντος `PATH` για να βρουν το εκτελέσιμο αρχείο
  - Οι `execl` και `execlp` θέλουν στο `arg0` το όνομα του εκτελέσιμου αρχείου, στα `arg1`, ... , `argn` τα ορίσματά του και το τελευταίο όρισμά τους `NULL`
  - Οι `execv` και `execvp` θέλουν το όνομα του εκτελέσιμου αρχείου και τα ορίσματά του στα `argv[0]`, `argv[1]`, ... , `argv[n]` και `NULL` στο `argv[n+1]`
  - Όλες οι κλήσεις της ομάδας `exec` επιστρέφουν λάθος (-1) αν δεν βρεθεί το εκτελέσιμο αρχείο, ενώ σε επιτυχία ποτέ δεν επιστρέφουν
  - Απαίτηση: `#include <unistd.h>`

- Χρήση της κλήσης `execl`

```
/* File: execl_demo.c */
#include <stdio.h>                          /* For printf */
#include <unistd.h>                          /* For execl */
main()
{  int ret;
   printf("I am process %d and I will execute an 'ls -l ..'\n",
          getpid());
   ret = execl("/bin/ls", "ls", "-l", "..", NULL);
   if (ret == -1) {  /* Always true because of failure
                              to execute /bin/ls -l .. */
      perror("execl"); } }
```

```
$ ./execl_demo
I am process 1614 and I will execute an 'ls -l ..'
total 12
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
drwxr-xr-x    2 spro     users        4096 Feb 15 22:24 c_progs
drwxr-xr-x    2 spro     users        4096 Feb 12 15:12 sh_scripts
$
```

- Χρήση της κλήσης execvp

```
/* File: execvp_demo.c */
#include <stdio.h>                          /* For printf */
#include <stdlib.h>                           /* For exit */
#include <unistd.h>                         /* For execvp */
main()
{  int pid, status; char *argv[2];
   if ((pid = fork()) == -1) {      /* Check for error */
      perror("fork");
      exit(1); }
   if (pid != 0) {                    /* Parent process */
      printf("I am parent process %d\n", getpid());
      if (wait(&status) != pid) {    /* Wait for child */
         perror("wait");
         exit(1); }
      printf("Child terminated with exit code %d\n",
             status >> 8); }
   else {                              /* Child process */
      argv[0] = "date";
      argv[1] = NULL;
      printf("I am child process %d", getpid());
      printf(" and I will replace myself by 'date'\n");
      execvp("date", argv);            /* Execute date */
      perror("execvp");
      exit(1); } }
```

```
$ ./execvp_demo
I am parent process 1615
I am child process 1616 and I will replace myself by 'date'
Fri Feb 15 22:26:26 EET 2002
Child terminated with exit code 0
$
```

- *Να γραφεί ένα πρόγραμμα C που να λειτουργεί σαν ένα απλό κέλυφος. Να δέχεται από το πληκτρολόγιο εντολές, τις οποίες να αναλαμβάνει να εκτελέσει μία διεργασία-παιδί που θα δημιουργεί.*

```
/* File: kind_of_shell.c */
#include <stdio.h>                      /* For fgets, printf */
#include <stdlib.h>                              /* For exit */
#include <unistd.h>                             /* For execvp */
void parse(char *buf, char **args);  /* Forward declaration */
void execute(char **args);           /* Forward declaration */

main()
{  char buf[1024]; char *args[64];
   for (;;) {
      printf("Command: ");                       /* Get input */
      if (fgets(buf, sizeof(buf), stdin) == NULL) {
         printf("\n"); exit(0); }
      parse(buf, args);       /* Split input into arguments */
      execute(args); } }             /* Execute the command */

void parse(char *buf, char **args)
{  while (*buf != '\0') {
      while ((*buf == ' ') || (*buf == '\t') || (*buf == '\n'))
         *buf++ = '\0';     /* Strip whitespace, write null */
      *args++ = buf;                     /* Save the argument */
      while ((*buf != '\0') && (*buf != ' ') &&
             (*buf != '\t') && (*buf != '\n'))
         buf++; }                 /* Skip over the argument */
   *--args = NULL; }                    /* End of arguments */

void execute(char **args)
{  int pid, status;
   if ((pid = fork()) < 0) {              /* Create a child */
      perror("fork"); exit(1); }
   if (pid == 0) {                         /* Child process */
      execvp(*args, args); perror(*args); exit(1); }
   if (wait(&status) != pid) {            /* Parent process */
         perror("wait"); exit(1); } }
```

```
$ ./kind_of_shell
Command: date
Sat Feb 16 21:22:53 EET 2002
Command: ls /
bin   dev  home    lib         misc  opt   root  tmp  var
boot  etc  initrd  lost+found  mnt   proc  sbin  usr
Command: whoami
spro
Command: ps
  PID TTY          TIME CMD
 2133 pts/0    00:00:00 bash
 7596 pts/0    00:00:00 kind_of_shell
 7600 pts/0    00:00:00 ps
Command: ls -l ..
total 12
drwxr-xr-x    2 spro     users        4096 Jan 27 12:56 bin
drwxr-xr-x    2 spro     users        4096 Feb 15 22:27 c_progs
drwxr-xr-x    2 spro     users        4096 Feb 12 15:12 sh_scripts
Command: pwd
/home/spro/c_progs
Command: echo "This is a test line"
"This is a test line"
Command: touch tempfile
Command: ls -l tempfile
-rw-r--r--    1 spro     users           0 Feb 16 21:24 tempfile
Command: rm tempfile
Command: rm *.*
rm: cannot remove `*.*': No such file or directory
Command: ^D
$
```

- *Να γραφεί ένα πρόγραμμα C που να δημιουργεί ένα δυαδικό δέντρο από διεργασίες το οποίο να έχει δεδομένο βάθος N. Κάθε διεργασία που δεν είναι φύλλο του δέντρου να εκτυπώνει την ταυτότητά της, την ταυτότητα του γονέα της καθώς και έναν αύξοντα αριθμό σύμφωνα με μία πρώτα κατά πλάτος διάσχιση του δέντρου.*

```
/* File: process_tree.c */
#include <stdio.h>                              /* For printf */
#include <stdlib.h>                               /* For exit */
main(int argc, char *argv[])
{  int i, depth, numb, pid1, pid2, status;
   if (argc > 1) depth = atoi(argv[1]);    /* Make integer */
   else          exit(0);
   if (depth > 5) {                    /* Avoid deep trees */
      printf("Depth should be up to 5\n"); exit(0); }
   numb = 1;           /* Holds the number of each process */
   for(i=1 ; i<=depth ; i++) {
      printf("I am process no %2d with PID %5d and PPID %5d\n",
             numb, getpid(), getppid());
      switch (pid1 = fork()) {
         case 0:                         /* Left child code */
            numb = 2*numb; break;
         case -1:             /* Error creating left child */
            perror("fork"); exit(1);
         default:                              /* Parent code */
            switch (pid2 = fork()) {
               case 0:                 /* Right child code */
                  numb = 2*numb+1; break;
               case -1:      /* Error creating right child */
                  perror("fork"); exit(1);
               default:                      /* Parent code */
                  wait(&status); wait(&status);
                  exit(0); } } } }
```

```
$ ./process_tree 1
I am process no  1 with PID  7608 and PPID  2133
$ ./process_tree 2
I am process no  1 with PID  7611 and PPID  2133
I am process no  2 with PID  7612 and PPID  7611
I am process no  3 with PID  7613 and PPID  7611
$ ./process_tree 3
I am process no  1 with PID  7618 and PPID  2133
I am process no  2 with PID  7619 and PPID  7618
I am process no  3 with PID  7620 and PPID  7618
I am process no  6 with PID  7623 and PPID  7620
I am process no  4 with PID  7621 and PPID  7619
I am process no  7 with PID  7624 and PPID  7620
I am process no  5 with PID  7622 and PPID  7619
$ ./process_tree 4
I am process no  1 with PID  7633 and PPID  2133
I am process no  2 with PID  7634 and PPID  7633
I am process no  4 with PID  7636 and PPID  7634
I am process no  8 with PID  7638 and PPID  7636
I am process no  3 with PID  7635 and PPID  7633
I am process no  7 with PID  7643 and PPID  7635
I am process no 14 with PID  7644 and PPID  7643
I am process no  6 with PID  7642 and PPID  7635
I am process no 12 with PID  7646 and PPID  7642
I am process no  9 with PID  7639 and PPID  7636
I am process no  5 with PID  7637 and PPID  7634
I am process no 10 with PID  7649 and PPID  7637
I am process no 11 with PID  7657 and PPID  7637
I am process no 13 with PID  7655 and PPID  7642
I am process no 15 with PID  7654 and PPID  7643
$
```

Αποστολή/παραλαβή σημάτων

- Από τον πυρήνα του Unix σε μία διεργασία, σε περίπτωση εξαιρετικού γεγονότος (λάθος κινητής υποδιαστολής, αντικανονική εντολή κ.λ.π.)
- Από το πληκτρολόγιο σε μία διεργασία για οριστική (`Control-C`) ή προσωρινή (`Control-Z`) διακοπή κ.λ.π.
- Από μία διεργασία σε μία άλλη διεργασία
- Η διεργασία που παραλαμβάνει ένα σήμα μπορεί είτε να το αγνοήσει είτε να διακόψει προσωρινά τη λειτουργία της για να εκτελέσει ένα τμήμα ειδικού κώδικα που ονομάζεται διαχειριστής του σήματος

- Στο Unix υπάρχουν διάφορα σήματα, κάθε ένα από τα οποία χαρακτηρίζεται από έναν αριθμό, ή από ένα ισοδύναμο συμβολικό όνομα, και έχει έναν προκαθορισμένο διαχειριστή σήματος, ο οποίος στις περισσότερες περιπτώσεις μπορεί να αλλάξει
- Σε προγράμματα που χρησιμοποιούν συμβολικά ονόματα σημάτων απαιτείται: `#include <signal.h>`
- Μερικά σήματα

`SIGINT`	(2)	interrupt
`SIGKILL`	(9)	kill
`SIGUSR1`	(10)	user-defined signal 1
`SIGUSR2`	(12)	user-defined signal 2
`SIGALRM`	(14)	alarm clock
`SIGTERM`	(15)	software termination signal
`SIGCONT`	(18)	continue after stop
`SIGSTOP`	(19)	stop
`SIGTSTP`	(20)	stop signal from keyboard

- Συνάρτηση βιβλιοθήκης `alarm`
  - `unsigned int alarm(unsigned int count)`
  - Δίνει εντολή στον πυρήνα να στείλει μετά από `count` δευτερόλεπτα το σήμα `SIGALRM` στην καλούσα διεργασία
  - Ο προκαθορισμένος διαχειριστής σήματος εκτυπώνει ένα μήνυμα και τερματίζει τη διεργασία
  - Αν η συνάρτηση κληθεί με `count` ίσο με 0, ακυρώνεται οποιοσδήποτε προηγούμενος προγραμματισμός, αν υπάρχει
  - Επιστρέφει τον χρόνο που απέμενε μέχρι να έλθει το σήμα που είχε προγραμματισθεί προηγουμένως, ή 0 αν δεν υπήρχε προγραμματισμός
- Χρήση της συνάρτησης `alarm`

```
/* File: alarm_demo.c */
#include <stdio.h>                  /* For printf */
main()
{  alarm(3);          /* Schedule an alarm signal
                                  in three seconds */
   printf("Looping forever...\n");
   while (1);
   printf("This line should never be executed\n"); }
```

```
$ date ; ./alarm_demo ; date
Sat Feb 16 21:40:32 EET 2002
Looping forever...
Alarm clock
Sat Feb 16 21:40:35 EET 2002
$
```

- Κλήση συστήματος `signal`
  - `void (*signal(int sigCode, void (*func)(int)))(int)`
  - Ορίζει την αντίδραση της καλούσας διεργασίας για το σήμα `sigCode`
  - Το δεύτερο όρισμα μπορεί να είναι `SIG_IGN`, οπότε το σήμα αγνοείται, `SIG_DFL`, που σημαίνει ότι ο προκαθορισμένος από το Unix διαχειριστής σήματος πρέπει να χρησιμοποιηθεί, ή η διεύθυνση μίας συνάρτησης που ορίζεται από τον προγραμματιστή και παίζει το ρόλο του διαχειριστή σήματος
  - Όταν καλείται ο διαχειριστής σήματος, το όρισμά του είναι ο αριθμός του σήματος που τον ενεργοποίησε
  - Τα σήματα `SIGKILL` και `SIGSTOP` δεν είναι δυνατόν να αγνοηθούν ούτε μπορεί να αλλάξει γι΄ αυτά ο διαχειριστής τους
  - Ό,τι ισχύει σε σχέση με τα σήματα για μία διεργασία κληρονομείται στις διεργασίες-παιδιά που δημιουργεί μέσω `fork`
  - Η `signal` επιστρέφει τον προηγούμενο διαχειριστή σήματος σε επιτυχία ή -1 σε αποτυχία

- Χρήση της κλήσης `signal`

```
/* File: critical.c */
#include <stdio.h>                              /* For printf */
#include <signal.h>                   /* For SIGINT, SIG_IGN */
main()
{  void (*oldHandler)(int);   /* To hold old handler value */
   printf("I can be Control-C'ed\n");
   sleep(3);
   oldHandler = signal(SIGINT, SIG_IGN);      /* Ignore ^C */
   printf("I am protected from Control-C now\n");
   sleep(3);
   signal(SIGINT, oldHandler);      /* Restore old handler */
   printf("I can be Control-C'ed again\n");
   sleep(3);
   printf("Bye!\n"); }
```

```
$ ./critical
I can be Control-C'ed
^C
$ ./critical
I can be Control-C'ed
I am protected from Control-C now
^CI can be Control-C'ed again
Bye!
$
```

- Κλήση συστήματος pause
  - `int pause()`
  - Θέτει την καλούσα διεργασία σε αναμονή μέχρις ότου ληφθεί κάποιο σήμα
- Χρήση της κλήσης pause

```
/* File: new_handler.c */
#include <stdio.h>                              /* For printf */
#include <signal.h>                            /* For SIGALRM */
int alarmFlag = 0;                        /* Global alarm flag */
void alarmHandler(int);                 /* Forward declaration */

main()
{  signal(SIGALRM, alarmHandler); /* Install signal handler */
   alarm(3);   /* Schedule an alarm signal in three seconds */
   printf("Looping...\n");
   while (!alarmFlag)                  /* Loop until flag set */
      pause();                           /* Wait for a signal */
   printf("Loop ends due to alarm signal\n"); }

void alarmHandler(int sig)
{  printf("An alarm clock signal (= %d) was received\n", sig);
   alarmFlag = 1; }              /* Set flag to stop looping */
```

```
$ date ; ./new_handler ; date
Sat Feb 16 21:54:30 EET 2002
Looping...
An alarm clock signal (= 14) was received
Loop ends due to alarm signal
Sat Feb 16 21:54:33 EET 2002
$
```

- Κλήση συστήματος `kill`
  - `int kill(int pid, int sigCode)`
  - Στέλνει το σήμα `sigCode` στη διεργασία με ταυτότητα `pid`
  - Είναι επιτυχής όταν η αποστέλλουσα διεργασία και η παραλαμβάνουσα διεργασία έχουν τον ίδιο ιδιοκτήτη ή όταν η αποστέλλουσα διεργασία ανήκει στο διαχειριστή του συστήματος
  - Επιστρέφει 0 σε επιτυχία ή -1 σε αποτυχία

- Χρήση της κλήσης `kill`

```
/* File: pulse.c */
#include <stdio.h>                                  /* For printf */
#include <signal.h>          /* For SIGTERM, SIGSTOP, SIGCONT */
#include <stdlib.h>                                   /* For exit */
main()
{  int pid1, pid2;
   if ((pid1 = fork()) == -1) {            /* Check for error */
      perror("fork"); exit(1); }
   if (pid1 == 0)                             /* First child */
      while (1) {                            /* Infinite loop */
         printf("Process 1 is alive\n"); sleep(1); }
   if ((pid2 = fork()) == -1) {            /* Check for error */
      perror("fork"); exit(1); }
   if (pid2 == 0)                              /* Second child */
      while (1) {                            /* Infinite loop */
         printf("Process 2 is alive\n"); sleep(1); }
   sleep(2); kill(pid1, SIGSTOP);      /* Suspend first child */
   sleep(2); kill(pid1, SIGCONT);       /* Resume first child */
   sleep(2);
   kill(pid1, SIGTERM);              /* Terminate first child */
   kill(pid2, SIGTERM); }           /* Terminate second child */
```

```
$ ./pulse
Process 1 is alive
Process 2 is alive
Process 1 is alive
Process 2 is alive
Process 2 is alive
Process 2 is alive
Process 1 is alive
Process 2 is alive
Process 1 is alive
Process 2 is alive
$
```

- *Να γραφεί ένα πρόγραμμα C το οποίο να βάζει ένα χρονικό όριο στην εκτέλεση εντολών.*

```
/* File: timeout.c */
#include <stdio.h>                          /* For printf */
#include <stdlib.h>                           /* For exit */
#include <unistd.h>                         /* For execvp */
#include <signal.h>                /* For SIGALRM, SIGKILL */
int pid;                                /* Global child pid */
void onalarm(int);                 /* Forward declaration */

main(int argc, char *argv[])
{  int sec = 10, status; char *progname;
   progname = argv[0];
   if (argc > 1 && *argv[1] == '-') { /* Optional timeout */
      sec = atoi(argv[1]+1);        /* Convert to integer */
      argc--;
      argv++; }
   if (argc < 2) {                     /* No command given */
      printf("Usage: %s [-10] command\n", progname);
      exit(1); }
   if ((pid = fork()) < 0) {           /* Check for error */
      perror("fork"); exit(1); }
   if (pid == 0) {                       /* Child process */
      execvp(argv[1], &argv[1]);       /* Execute command */
      perror("execvp"); exit(1); }
   signal(SIGALRM, onalarm);    /* Install signal handler */
   alarm(sec);                         /* Schedule an alarm */
   wait(&status);          /* Wait for child to terminate */
   if ((status & 0377) == 0) {
      printf("Command execution finished");
      printf(" with exit code %d\n", status >> 8); }
   else
      printf("Command was killed by signal %d\n",
             status & 0177); }

void onalarm(int sig)
{  kill(pid, SIGKILL); }   /* Timeout for child - Kill it */
```

```
$ ./timeout
Usage: ./timeout [-10] command
$ ./timeout -30
Usage: ./timeout [-10] command
$ ./timeout blabla
execvp: No such file or directory
Command execution finished with exit code 1
$ ./timeout -20 ps
  PID TTY          TIME CMD
12947 pts/0    00:00:00 bash
13026 pts/0    00:00:00 timeout
13027 pts/0    00:00:00 ps
Command execution finished with exit code 0
$ date ; ./timeout -12 sleep 15 ; date
Mon Feb 18 18:43:20 EET 2002
Command was killed by signal 9
Mon Feb 18 18:43:32 EET 2002
$ date ; ./timeout sleep 7 ; date
Mon Feb 18 18:43:49 EET 2002
Command execution finished with exit code 0
Mon Feb 18 18:43:56 EET 2002
$
```

- *Να γραφεί ένα πρόγραμμα C το οποίο, σαν διεργασία, όταν δέχεται ένα σήμα (για παράδειγμα το SIGUSR1) να δημιουργεί ένα αντίγραφο του εαυτού του (διεργασία-παιδί) και μετά να τερματίζει.*

```
/* File: persistent.c */
#include <stdio.h>                          /* For printf */
#include <stdlib.h>                           /* For exit */
#include <signal.h>                         /* For SIGUSR1 */
int loop;
void sigusr1_handler(int);       /* Forward declaration */

main()
{  int pid;
   signal(SIGUSR1, sigusr1_handler); /* Install handler */
   while(1) {                              /* Loop forever */
      printf("I am process %d and I am going to loop\n",
             getpid());
      loop = 1;
      while(loop) {                /* Loop until loop=0 */
         sleep(1); }
      if ((pid = fork()) == -1) {    /* Check for error */
         perror("fork"); exit(1); }
      if (pid == 0)
         continue;                     /* Child process */
      else
         exit(0); } }                 /* Parent process */

void sigusr1_handler(int sig)
{  printf("SIGUSR1 (= %d) received\n", sig);
   loop = 0; }
```

```
$ ./persistent &
[1] 13059
I am process 13059 and I am going to loop
$ kill -USR1 13059
SIGUSR1 (= 10) received
I am process 13060 and I am going to loop
[1]+  Done                    ./persistent
$ kill -USR1 13060
$ SIGUSR1 (= 10) received
I am process 13061 and I am going to loop
kill -USR1 13061
SIGUSR1 (= 10) received
$ I am process 13062 and I am going to loop
kill -USR1 13062
$ SIGUSR1 (= 10) received
I am process 13063 and I am going to loop
kill -KILL 13063
$
```

Είσοδος/έξοδος χαμηλού επιπέδου

- Για απλές διαδικασίες εισόδου/εξόδου χρησιμοποιείται η προκαθορισμένη βιβλιοθήκη `stdio`, η οποία παρέχει σημαντικές ευκολίες, όπως ενδιάμεση μνήμη και μετατροπή δεδομένων
- Η `stdio` είναι ένα φιλικό εργαλείο για τον προγραμματιστή και έχει κατασκευασθεί μέσω ενός συνόλου από κλήσεις συστήματος που υποστηρίζουν είσοδο/έξοδο χαμηλού επιπέδου
- Οι κλήσεις συστήματος για είσοδο/έξοδο χαμηλού επιπέδου είναι αναγκαίες μόνο όταν οι ευκολίες που παρέχει η `stdio` για προσπέλαση αρχείων ή συσκευών δεν είναι επιθυμητές ή όταν προγραμματίζεται επικοινωνία μεταξύ διεργασιών μέσω σωλήνων και υποδοχών

- Ο προσδιορισμός αρχείων, συσκευών, άκρων σωλήνων και υποδοχών γίνεται στις κλήσεις συστήματος για είσοδο/έξοδο χαμηλού επιπέδου από περιγραφείς αρχείων που είναι ακέραιοι αριθμοί, σε αντίθεση με την `stdio` όπου χρησιμοποιούνται, για αρχεία και συσκευές, δείκτες σε αρχεία
- Προκαθορισμένοι περιγραφείς αρχείων

  `0` : Προκαθορισμένη είσοδος

  `1` : Προκαθορισμένη έξοδος

  `2` : Προκαθορισμένη έξοδος για διαγνωστικά μηνύματα
- Οι προκαθορισμένοι περιγραφείς αρχείων `0`, `1` και `2` αντιστοιχούν στους προκαθορισμένους δείκτες σε αρχεία `stdin`, `stdout` και `stderr` της `stdio`
- Οι περιγραφείς αρχείων κληρονομούνται από μία διεργασία-γονέα στις διεργασίες-παιδιά που δημιουργεί

- Κλήση συστήματος `open`
  - `int open(char *fileName, int mode [, int permissions])`
  - Ανοίγει ή δημιουργεί το αρχείο με απόλυτο ή σχετικό όνομα `fileName` για διάβασμα και/ή γράψιμο
  - Το `mode` είναι ένας ακέραιος που παριστάνει τον τρόπο προσπέλασης του αρχείου και δίνεται σαν διάζευξη συμβολικών ονομάτων, όπως:

    `O_RDONLY`: Ανοιγμα για διάβασμα μόνο
    `O_WRONLY`: Ανοιγμα για γράψιμο μόνο
    `O_RDWR`: Ανοιγμα για διάβασμα και γράψιμο
    `O_APPEND`: Γράψιμο στο τέλος του αρχείου
    `O_CREAT`: Δημιουργία αρχείου, εφ΄ όσον δεν υπάρχει
    `O_TRUNC`: Διαγραφή περιεχομένων αρχείου, εφ΄ όσον υπάρχει

    Απαίτηση: `#include <fcntl.h>`
  - Το προαιρετικό όρισμα `permissions` είναι ένας ακέραιος του οποίου η τιμή πρέπει να αντιστοιχεί στα επιθυμητά δικαιώματα προστασίας ενός αρχείου κατά τη δημιουργία του (δικαιώματα που αποκλείονται από την τρέχουσα τιμή του `umask` δεν δίνονται στο αρχείο)
  - Η `open` επιστρέφει έναν περιγραφέα αρχείου σε επιτυχία ή -1 σε αποτυχία

- Κλήση συστήματος `read`
  - `int read(int fd, char *buf, int count)`
  - Διαβάζει το πολύ `count` bytes από την οντότητα (αρχείο, συσκευή, άκρο σωλήνα ή υποδοχή) που αντιστοιχεί στον περιγραφέα `fd` και τα τοποθετεί στο `buf`
  - Επιστρέφει τον αριθμό των bytes που διαβάστηκαν, 0 αν έγινε απόπειρα διαβάσματος αφού έχει διαβαστεί και το τελευταίο byte ή -1 σε αποτυχία
- Κλήση συστήματος `write`
  - `int write(int fd, char *buf, int count)`
  - Γράφει το πολύ `count` bytes από το `buf` στην οντότητα που αντιστοιχεί στον περιγραφέα `fd`
  - Επιστρέφει τον αριθμό των bytes που γράφτηκαν ή -1 σε αποτυχία
- Κλήση συστήματος `close`
  - `int close(int fd)`
  - Ελευθερώνει τον περιγραφέα `fd`
  - Επιστρέφει 0 σε επιτυχία ή -1 σε αποτυχία

- Χρήση των κλήσεων open, read, write και close

```
/* File: io_demo.c */
#include <stdio.h>                          /* For printf */
#include <stdlib.h>                           /* For exit */
#include <fcntl.h>              /* For O_RDONLY, O_WRONLY,
                                        O_CREAT, O_APPEND */
main()
{  int fd, bytes;
   char buf[50];
   if ((fd = open("t", O_WRONLY | O_CREAT, 0600)) == -1) {
      perror("open"); exit(1); }
   bytes = write(fd, "First write. ", 13);   /* Data out */
   printf("%d bytes were written\n", bytes);
   close(fd);
   if ((fd = open("t", O_WRONLY | O_APPEND)) == -1) {
      perror("open"); exit(1); }
   bytes = write(fd, "Second write.\n", 14); /* Data out */
   printf("%d bytes were written\n", bytes);
   close(fd);
   if ((fd = open("t", O_RDONLY)) == -1) {
      perror("open"); exit(1); }
   bytes = read(fd, buf, sizeof(buf));        /* Data in */
   printf("%d bytes were read\n", bytes);
   close(fd);
   buf[bytes] = '\0';
   printf("%s", buf); }
```

```
$ ./io_demo
13 bytes were written
14 bytes were written
27 bytes were read
First write. Second write.
$ ls -l t
-rw-------    1 spro     users          27 Feb 18 19:28 t
$ rm t
$
```

- Κλήσεις συστήματος `dup` και `dup2`
  - `int dup(int oldFd)`
  - `int dup2(int oldFd, int newFd)`
  - Η `dup` βρίσκει το μικρότερο ελεύθερο περιγραφέα αρχείου και τον αντιστοιχεί στην ίδια οντότητα με τον περιγραφέα `oldFd`
  - Η `dup2` ελευθερώνει τον περιγραφέα `newFd`, εφ΄ όσον είναι δεσμευμένος, και τον αντιστοιχεί στην ίδια οντότητα με τον περιγραφέα `oldFd`
  - Σε επιτυχία επιστρέφουν το νέο περιγραφέα, δηλαδή η `dup2` το `newFd`, ή -1 σε αποτυχία

- Χρήση των κλήσεων dup και dup2

```
/* File: duplicate_fd.c */
#include <stdio.h>                          /* For printf */
#include <stdlib.h>                           /* For exit */
#include <fcntl.h>      /* For O_RDWR, O_CREAT, O_TRUNC */
main()
{  int fd1, fd2, fd3;
   if ((fd1 = open("r", O_RDWR | O_CREAT | O_TRUNC,
                  0644)) == -1) {
      perror("open");
      exit(1); }
   printf("fd1 = %d\n", fd1);
   write(fd1, "What ", 5);          /* Write data to fd1 */
   fd2 = dup(fd1);                 /* Make a copy of fd1 */
   printf("fd2 = %d\n", fd2);
   write(fd2, "time", 4);           /* Write data to fd2 */
   close(0);                      /* Close standard input */
   fd3 = dup(fd1);                /* Another copy of fd1 */
   printf("fd3 = %d\n", fd3);
   write(fd3, " is it", 6);         /* Write data to fd3 */
   dup2(fd1, 2);                  /* Duplicate fd1 to 2 */
   write(2, "?\n", 2);                /* Write data to 2 */
   close(fd1);
   close(fd2);
   close(fd3); }
```

```
$ ./duplicate_fd
fd1 = 3
fd2 = 4
fd3 = 0
$ ls -l r
-rw-r--r--    1 spro     users          17 Feb 18 19:45 r
$ cat r
What time is it?
$ rm r
$
```

- *Να γραφεί ένα πρόγραμμα C που να αντιγράφει ένα αρχείο στο τέλος ενός άλλου.*

```
/* File: append_file.c */
#include <string.h>                     /* For strlen */
#include <stdlib.h>                       /* For exit */
#include <fcntl.h>          /* For O_RDONLY, O_WRONLY,
                                      O_CREAT, O_APPEND */
main(int argc, char *argv[])
{  int n, from, to;
   char buf[1024];
   if (argc != 3) {         /* Check for proper usage */
      write(2, "Usage: ", 7);
      write(2, *argv, strlen(*argv));
      write(2, " from-file to-file\n", 19);
      exit(1); }
   if ((from = open(argv[1], O_RDONLY)) < 0) {
                                    /* Open from-file */
      perror("open");
      exit(1); }
   if ((to = open(argv[2], O_WRONLY | O_CREAT | O_APPEND,
                 0660)) < 0) {       /* Open to-file */
      perror("open");
      exit(1); }
   while ((n = read(from, buf, sizeof(buf))) > 0)
      write(to, buf, n);                 /* Copy data */
   close(from);                     /* Close from-file */
   close(to); }                       /* Close to-file */
```

```
$ ./append_file
Usage: ./append_file from-file to-file
$ ls -l file*
-rw-r--r--    1 spro     users          39 Feb 18 19:51 file1
-rw-r--r--    1 spro     users          26 Feb 18 19:52 file2
$ cat file1
file1 line 1
file1 line 2
file1 line 3
$ cat file2
file2 line 1
file2 line 2
$ ./append_file file2 file1
$ cat file1
file1 line 1
file1 line 2
file1 line 3
file2 line 1
file2 line 2
$ ./append_file file2 file3
$ cat file3
file2 line 1
file2 line 2
$ rm file1 file2 file3
$
```

Επικοινωνία μεταξύ διεργασιών

- Μέσω σωλήνων
  - Για διεργασίες που έχουν κοινό πρόγονο ο οποίος έχει δημιουργήσει το σωλήνα
  - Επικοινωνία προς μία κατεύθυνση μόνο
  - Μία διεργασία γράφει στο άκρο γραψίματος του σωλήνα και η άλλη διαβάζει από το άκρο διαβάσματος
- Μέσω υποδοχών
  - Μεταξύ διεργασιών που οι ίδιες δημιουργούν τις υποδοχές
  - Επικοινωνία και προς τις δύο κατευθύνσεις
  - Οι διεργασίες μπορεί να εκτελούνται στον ίδιο υπολογιστή ή ακόμα και σε διαφορετικούς υπολογιστές που είναι συνδεδεμένοι μέσω ενός δικτύου

- Κλήση συστήματος `pipe`
  - `int pipe(int fd[])`
  - Δημιουργεί ένα σωλήνα με άκρο διαβάσματος που αντιστοιχεί στον περιγραφέα `fd[0]` και άκρο γραψίματος που αντιστοιχεί στον περιγραφέα `fd[1]`
  - Απόγονοι της διεργασίας που δημιούργησε το σωλήνα, συμπεριλαμβανομένης και της ίδιας της διεργασίας-δημιουργού, μπορούν να επικοινωνήσουν μέσω του σωλήνα, που αποτελεί ουσιαστικά ένα είδος ενδιάμεσης μνήμης, γραφοντας με τη `write` και διαβάζοντας με τη `read` από το κατάλληλο άκρο
  - Τα μη χρησιμοποιούμενα άκρα πρέπει να ελευθερώνονται με την `close`
  - Η `pipe` επιστρέφει 0 σε επιτυχία ή -1 σε αποτυχία

- Χρήση της κλήσης pipe

```
/* File: pipe_demo.c */
#include <stdio.h>                    /* For printf */
#include <string.h>                   /* For strlen */
#include <stdlib.h>                     /* For exit */
#define READ  0                  /* Read end of pipe */
#define WRITE 1                 /* Write end of pipe */
char *phrase = "This is a test phrase.";
main()
{  int pid, fd[2], bytes;
   char message[100];
   if (pipe(fd) == -1) {            /* Create a pipe */
      perror("pipe");
      exit(1); }
   if ((pid = fork()) == -1) {       /* Fork a child */
      perror("fork");
      exit(1); }
   if (pid == 0) {                  /* Child, writer */
      close(fd[READ]);           /* Close unused end */
      write(fd[WRITE], phrase, strlen(phrase)+1);
      close(fd[WRITE]); }          /* Close used end */
   else {                           /* Parent, reader */
      close(fd[WRITE]);          /* Close unused end */
      bytes = read(fd[READ], message, sizeof(message));
      printf("Read %d bytes: %s\n", bytes, message);
      close(fd[READ]); } }         /* Close used end */
```

```
$ ./pipe_demo
Read 23 bytes: This is a test phrase.
$
```

- *Να γραφεί ένα πρόγραμμα C που να συνδέει μέσω ενός σωλήνα την προκαθορισμένη έξοδο μίας εντολής με την προκαθορισμένη είσοδο μίας άλλης, να υλοποιηθεί δηλαδή μία σωλήνωση.*

```
/* File: connect.c */
#include <stdio.h>                          /* For printf */
#include <stdlib.h>                           /* For exit */
#include <unistd.h>                         /* For execlp */
#define READ  0                          /* Read end of pipe */
#define WRITE 1                         /* Write end of pipe */
main(int argc, char *argv[])
{  int fd[2], pid;
   if (pipe(fd) == -1) {                  /* Create a pipe */
      perror("pipe");
      exit(1); }
   if ((pid = fork()) == -1) {             /* Fork a child */
      perror("fork");
      exit(1); }
   if (pid != 0) {                       /* Parent, writer */
      close(fd[READ]);                  /* Close unused end */
      dup2(fd[WRITE], 1); /* Duplicate write end to stdout */
      close(fd[WRITE]);                  /* Close write end */
      execlp(argv[1], argv[1], NULL);   /* Execute argv[1] */
      perror("execlp"); }
   else {                                  /* Child, reader */
      close(fd[WRITE]);                 /* Close unused end */
      dup2(fd[READ], 0);    /* Duplicate read end to stdin */
      close(fd[READ]);                   /* Close read end */
      execlp(argv[2], argv[2], NULL);   /* Execute argv[2] */
      perror("execlp"); } }
```

```
$ ./connect who wc
      1       6      51
$ ./connect ls wc
     55      55     715
$ ./connect ls head
alarm_demo
alarm_demo.c
append_file
append_file.c
connect
connect.c
critical
critical.c
duplicate_fd
duplicate_fd.c
$ ./connect ps sort
$ 12947 pts/0    00:00:00 bash
13236 pts/0    00:00:00 ps
13237 pts/0    00:00:00 sort
  PID TTY          TIME CMD

$
```

- Πεδία υποδοχών
  - `PF_INET`
    - Πεδίο Internet
    - Επικοινωνία και σε διαφορετικούς υπολογιστές
    - Διεύθυνση: Internet διεύθυνση και αριθμός θύρας
  - `PF_UNIX`
    - Πεδίο Unix
    - Επικοινωνία στον ίδιο υπολογιστή
    - Διεύθυνση: Κόμβος στο σύστημα αρχείων
- Είδη υποδοχών
  - `SOCK_STREAM`
    - Υποδοχές ροής (TCP)
  - `SOCK_DGRAM`
    - Τηλεγραφικές υποδοχές (UDP)

	TCP	UDP
Απαίτηση σύνδεσης	ΝΑΙ	ΟΧΙ
Αξιοπιστία	ΝΑΙ	ΟΧΙ
Όρια μηνυμάτων	ΟΧΙ	ΝΑΙ
Διαδοχικότητα μηνυμάτων	ΝΑΙ	ΟΧΙ

- Συναρτήσεις βιβλιοθήκης `htons`, `htonl`, `ntohs` και `ntohl`
  - `unsigned short htons(unsigned short hostshort)`
  - `unsigned long htonl(unsigned long hostlong)`
  - `unsigned short ntohs(unsigned short netshort)`
  - `unsigned long ntohl(unsigned long netlong)`
  - Μετατροπή ακολουθίας bytes από διάταξη "μηχανής" σε διάταξη "δικτύου" και αντίστροφα για short και long ακεραίους
  - Απαιτήσεις
    - `#include <sys/types.h>`
    - `#include <netinet/in.h>`
- Συναρτήσεις βιβλιοθήκης `gethostbyname` και `gethostbyaddr`
  - `struct hostent *gethostbyname(char *name)`
  - `struct hostent *gethostbyaddr(char *addr, int len, int type)`
  - Επιστροφή ενός δείκτη σε δομή `struct hostent` για έναν υπολογιστή είτε δεδομένου του ονόματός του `name`, όπου στο πεδίο `h_addr` της δομής βρίσκεται η Internet διεύθυνσή του και στο πεδίο `h_length` το μέγεθός της, είτε δεδομένης της διεύθυνσής του `addr`, του μεγέθους της `len` και του είδους της `type` (πάντα `PF_INET`), όπου στο πεδίο `h_name` της επιστρεφόμενης δομής βρίσκεται το όνομα του υπολογιστή
  - Απαίτηση: `#include <netdb.h>`

- Όλες οι κλήσεις συστήματος που ακολουθούν και χρησιμοποιούνται για διαχείριση υποδοχών απαιτούν
  - `#include <sys/types.h>`
  - `#include <sys/socket.h>`

  και επιστρεφουν -1 σε περίπτωση αποτυχίας
- Κλήση συστήματος `socket`
  - `int socket(int domain, int type, int protocol)`
  - Δημιουργεί μία υποδοχή και επιστρέφει τον περιγραφέα αρχείου που αντιστοιχεί σ΄ αυτήν
  - Το `domain` πρέπει να είναι `PF_INET` ή `PF_UNIX`
  - Το `type` πρέπει να είναι `SOCK_STREAM` ή `SOCK_DGRAM`
  - Σαν `protocol`, πάντα δίνουμε το default (`0`)

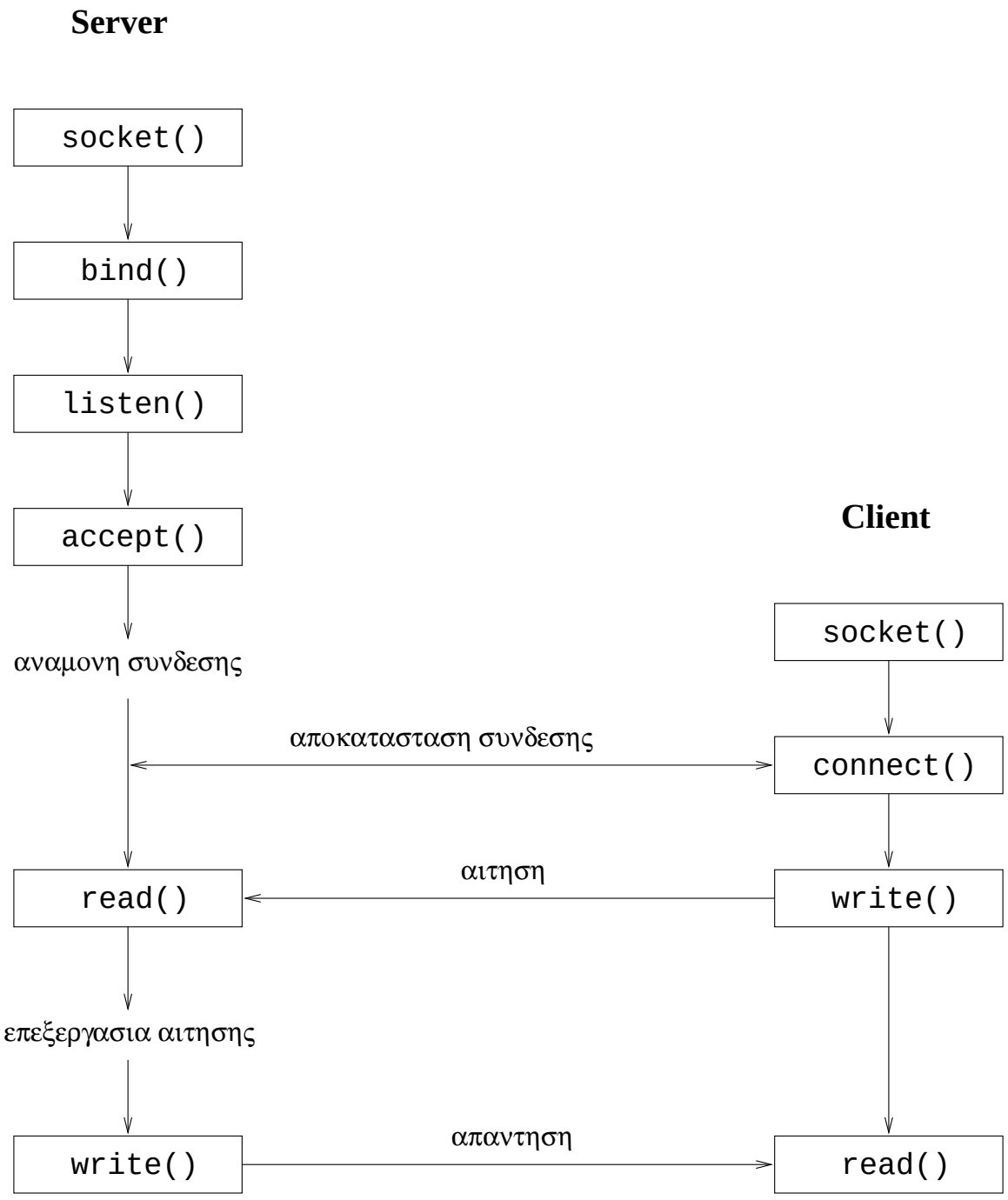
- Κλήση συστήματος `bind`
  - `int bind(int fd, struct sockaddr *address, unsigned int addresslen)`
  - Συνδέει την υποδοχή που αντιστοιχεί στον περιγραφέα αρχείου `fd` με ένα όνομα/διεύθυνση `*address`
  - Πεδίο Internet
    - Ορίζεται ένα `struct sockaddr_in name` και δίνονται τα `PF_INET` στο πεδίο `name.sin_family`, `htonl(INADDR_ANY)` στο πεδίο `name.sin_addr.s_addr` και `htons(port)` στο πεδίο `name.sin_port`, όπου `port` είναι ο αριθμός θύρας που χρησιμοποιείται και αφού η διεύθυνση του `name` προσαρμοσθεί σε `(struct sockaddr *)` δίνεται στο `address`
    - Απαίτηση: `#include <netinet/in.h>`
  - Πεδίο Unix
    - Ορίζεται ένα `struct sockaddr_un name` και δίνονται τα `PF_UNIX` στο πεδίο `name.sun_family` και `path` στο πεδίο `name.sun_path`, όπου `path` είναι το όνομα-μονοπάτι του κόμβου που χρησιμοποιείται στο σύστημα αρχείων και αφού η διεύθυνση του `name` προσαρμοσθεί σε `(struct sockaddr *)` δίνεται στο `address`
    - Απαίτηση: `#include <sys/un.h>`
  - Το μέγεθος του `name` δίνεται στο `addresslen`

- Κλήση συστήματος `listen`
  - `int listen(int fd, int queuelength)`
  - Ορίζει μία ουρά μήκους `queuelength` σε έναν server στην οποία μπορούν να συσσωρεύονται αιτήσεις από clients για σύνδεση στην υποδοχή που αντιστοιχεί στον περιγραφέα αρχείου `fd`
- Κλήση συστήματος `accept`
  - `int accept(int fd, struct sockaddr *address, unsigned int *addresslen)`
  - Αποδέχεται μία αίτηση σύνδεσης που έχει υποβληθεί σε έναν server στην υποδοχή με περιγραφέα αρχείου `fd`
  - Πληροφορίες για τη διεύθυνση του client που συνδέθηκε επιστρέφονται μέσω της δομής `*address` το μέγεθος της οποίας επιστρέφεται στο `*addresslen`
  - Επιστρέφει ένα νέο περιγραφέα αρχείου ο οποίος πρέπει να χρησιμοποιηθεί από τον server για επικοινωνία με τον client
- Κλήση συστήματος `connect`
  - `int connect(int fd, struct sockaddr *address, unsigned int addresslen)`
  - Υποβολή αίτησης σύνδεσης από έναν client μέσω υποδοχής που αντιστοιχεί στον περιγραφέα αρχείου `fd` με τον server του οποίου η διεύθυνση έχει κατασκευασθεί στη δομή `*address` το μέγεθος της οποίας έχει τεθεί στο `addresslen`

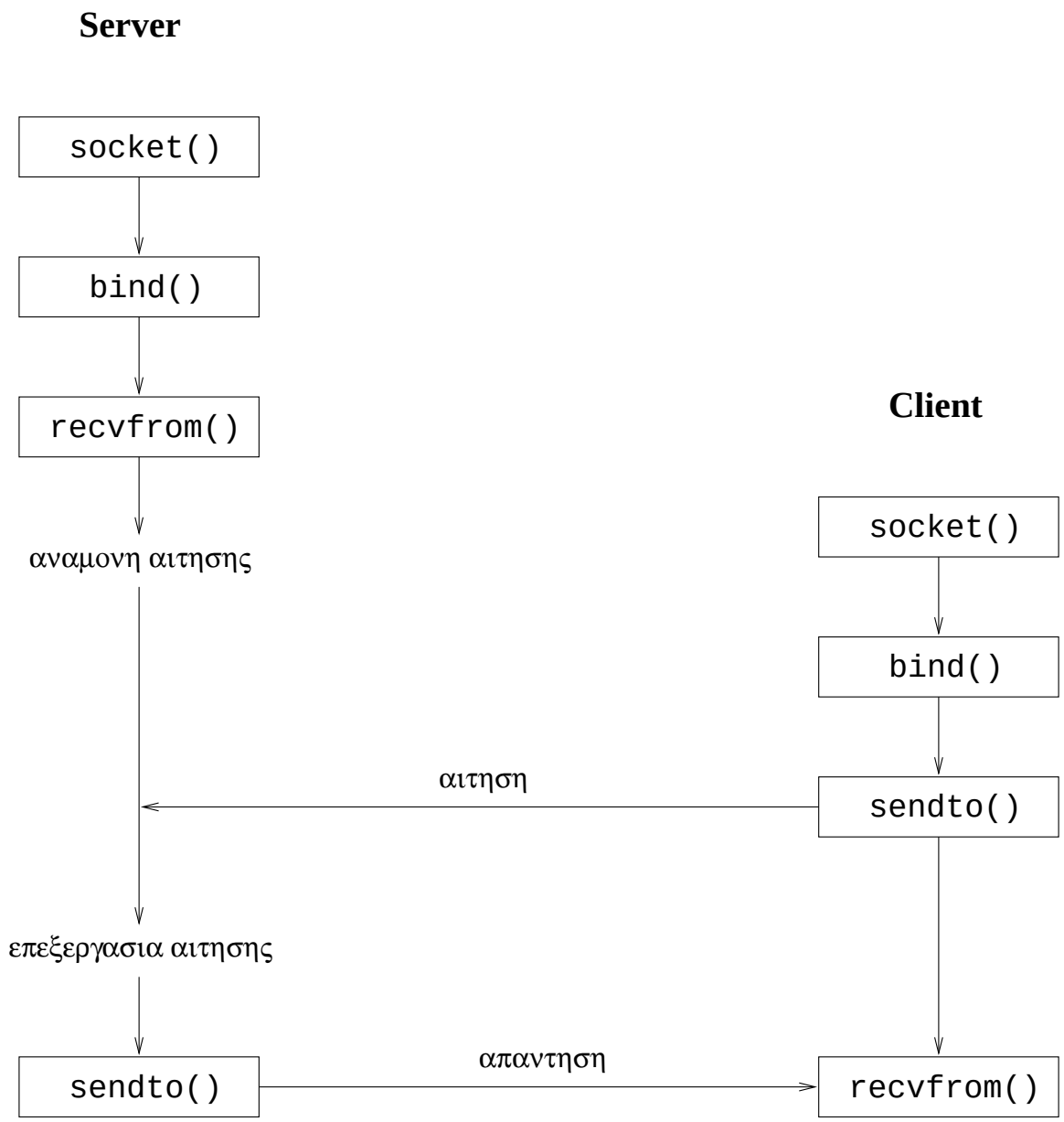
- Κλήσεις συστήματος `recvfrom` και `sendto`
  - `int recvfrom(int fd, char *buf, int count, int flags, struct sockaddr *address, unsigned int *addresslen)`
  - `int sendto(int fd, char *buf, int count, int flags, struct sockaddr *address, unsigned int addresslen)`
  - Χρησιμοποιούνται αντί των `read` και `write` για την παραλαβή και την αποστολή μηνυμάτων μέσω τηλεγραφικών υποδοχών
  - Τα `fd`, `buf` και `count` έχουν την ίδια σημασία όπως στις `read` και `write`
  - Στο `flags` συνήθως δίνεται η τιμή 0, εκτός αν πρόκειται να γίνει χειρισμός κάποιων ειδικών περιπτώσεων
  - Στη δομή `*address` επιστρέφεται η διεύθυνση της τηλεγραφικής υποδοχής που χρησιμοποιεί ο αποστολέας (για τη `recvfrom`) ή τίθεται η διεύθυνση της τηλεγραφικής υποδοχής που χρησιμοποιεί ο παραλήπτης (για τη `sendto`)
  - Στο `*addresslen` επιστρέφεται το μέγεθος της διεύθυνσης της υποδοχής του αποστολέα (για τη `recvfrom`), ενώ στο `addresslen` τίθεται η διεύθυνση της υποδοχής του παραλήπτη (για τη `sendto`)

- Κλήση συστήματος `getsockname`
  - `int getsockname(int fd,`
    `struct sockaddr *address,`
    `unsigned int *addresslen)`
  - Επιστρέφει στη δομή `*address` τη διεύθυνση με την οποία έχει συνδεθεί η υποδοχή που αντιστοιχεί στον περιγραφέα αρχείου `fd` και στο `*addresslen` το μέγεθος της διεύθυνσης αυτής
  - Είναι χρήσιμη στην περίπτωση που δεν προκαθορίζεται συγκεκριμένος αριθμός θύρας κατά τη σύνδεση μέσω της `bind` μίας υποδοχής με μία διεύθυνση (με απόδοση του 0 στο `name.sin_port`) για να βρεθεί ο αριθμός της πραγματικής θύρας που διατέθηκε από το σύστημα για τη σύνδεση
- Συναρτήσεις βιβλιοθήκης `bzero` και `bcopy`
  - `void bzero(char *buf, int count)`
    - Θέτει 0 σε `count` bytes αρχίζοντας από τη διεύθυνση `buf`
  - `void bcopy(char *buf1, char *buf2, int count)`
    - Αντιγράφει `count` bytes αρχίζοντας από τη διεύθυνση `buf1` στη διεύθυνση `buf2`
  - Απαίτηση: `#include <string.h>`
- Για μεταγλώττιση στο Solaris προγραμμάτων C με κλήσεις συστήματος για υποδοχές, πρέπει να προστίθεται στην εντολή μεταγλώττισης και το "`-lsocket -lnsl`"

- TCP επικοινωνία client/server



- UDP επικοινωνία client/server



- *Να γραφούν client και server προγράμματα C που να επιδεικνύουν την επικοινωνία μέσω υποδοχών ροής στο πεδίο Internet.*

```
/* File: int_str_server.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <netinet/in.h>                          /* For Internet sockets */
#include <netdb.h>                                  /* For gethostbyaddr */
#include <stdio.h>                                            /* For I/O */
#include <stdlib.h>                                          /* For exit */
#include <string.h>                                 /* For strlen, bzero */

void reverse(char *);

main(int argc, char *argv[])     /* Server with Internet stream sockets */
{  int port, sock, newsock; char buf[256];
   unsigned int serverlen, clientlen;
   struct sockaddr_in server, client;
   struct sockaddr *serverptr, *clientptr;
   struct hostent *rem;
   if (argc < 2) {            /* Check if server's port number is given */
      printf("Please give the port number\n");
      exit(1); }
   if ((sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) { /* Create socket */
      perror("socket"); exit(1); }
   port = atoi(argv[1]);              /* Convert port number to integer */
   server.sin_family = PF_INET;                      /* Internet domain */
   server.sin_addr.s_addr = htonl(INADDR_ANY);   /* My Internet address */
   server.sin_port = htons(port);                     /* The given port */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   if (bind(sock, serverptr, serverlen) < 0) {/* Bind socket to address */
      perror("bind"); exit(1); }
   if (listen(sock, 5) < 0) {                /* Listen for connections */
      perror("listen"); exit(1); }
   printf("Listening for connections to port %d\n", port);
```

```
   while(1) {
      clientptr = (struct sockaddr *) &client;
      clientlen = sizeof client;
      if ((newsock = accept(sock, clientptr, &clientlen)) < 0) {
         perror("accept"); exit(1); }              /* Accept connection */
      if ((rem = gethostbyaddr((char *) &client.sin_addr.s_addr,
            sizeof client.sin_addr.s_addr,     /* Find client's address */
            client.sin_family)) == NULL) {
         perror("gethostbyaddr"); exit(1); }
      printf("Accepted connection from %s\n", rem -> h_name);
      switch (fork()) {          /* Create child for serving the client */
         case -1:
            perror("fork"); exit(1);
         case 0:                                      /* Child process */
            do {
               bzero(buf, sizeof buf);             /* Initialize buffer */
               if (read(newsock, buf, sizeof buf) < 0) { /* Get message */
                  perror("read"); exit(1); }
               printf("Read string: %s\n", buf);
               reverse(buf);                          /* Reverse message */
               if (write(newsock, buf, sizeof buf) < 0){/* Send message */
                  perror("write"); exit(1); }
            } while (strcmp(buf, "dne") != 0);       /* Finish on "end" */
            close(newsock);                             /* Close socket */
            exit(0); } } }

void reverse(char *s)                /* Function for reversing a string */
{  int c, i, j;
   for (i = 0, j = strlen(s) - 1 ; i < j ; i++, j--) {
      c = s[i];
      s[i] = s[j];
      s[j] = c; } }
```

```
/* File: int_str_client.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <netinet/in.h>                          /* For Internet sockets */
#include <netdb.h>                                  /* For gethostbyname */
#include <stdio.h>                                            /* For I/O */
#include <stdlib.h>                                          /* For exit */
#include <string.h>                             /* For strlen, bzero, bcopy */
main(int argc, char *argv[])     /* Client with Internet stream sockets */
{  int port, sock; char buf[256]; unsigned int serverlen;
   struct sockaddr_in server;
   struct sockaddr *serverptr;
   struct hostent *rem;
   if (argc < 3) {     /* Are server's host name and port number given? */
      printf("Please give host name and port number\n"); exit(1); }
   if ((sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) { /* Create socket */
      perror("socket"); exit(1); }
   if ((rem = gethostbyname(argv[1])) == NULL) { /* Find server address */
      perror("gethostbyname"); exit(1); }
   port = atoi(argv[2]);              /* Convert port number to integer */
   server.sin_family = PF_INET;                       /* Internet domain */
   bcopy((char *) rem -> h_addr, (char *) &server.sin_addr,
         rem -> h_length);
   server.sin_port = htons(port); /* Server's Internet address and port */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   if (connect(sock, serverptr, serverlen) < 0) { /* Request connection */
      perror("connect"); exit(1); }
   printf("Requested connection to host %s port %d\n", argv[1], port);
   do {
      bzero(buf, sizeof buf);                      /* Initialize buffer */
      printf("Give input string: ");
      fgets(buf, sizeof buf, stdin);          /* Read message from stdin */
      buf[strlen(buf)-1] = '\0';             /* Remove newline character */
      if (write(sock, buf, sizeof buf) < 0) {           /* Send message */
         perror("write"); exit(1); }
      bzero(buf, sizeof buf);                      /* Initialize buffer */
      if (read(sock, buf, sizeof buf) < 0) {         /* Receive message */
         perror("read"); exit(1); }
      printf("Read string:       %s\n", buf);
   } while (strcmp(buf, "dne") != 0);                /* Finish on "end" */
   close(sock); exit(0); }                     /* Close socket and exit */
```

```
$ ./int_str_server 30000
Listening for connections to port 30000
Accepted connection from knossos.di.uoa.gr
Read string: test
Read string: A string
Read string: niconanomimatamimonanocin
Read string: This is a test line
Read string: Teleiosame
Read string: end
^C
$
```

```
$ ./int_str_client galini.di.uoa.gr 30000
Requested connection to host galini.di.uoa.gr port 30000
Give input string: test
Read string:       tset
Give input string: A string
Read string:       gnirts A
Give input string: niconanomimatamimonanocin
Read string:       niconanomimatamimonanocin
Give input string: This is a test line
Read string:       enil tset a si sihT
Give input string: Teleiosame
Read string:       emasoieleT
Give input string: end
Read string:       dne
$
```

- *Να γραφούν client και server προγράμματα C που να επιδεικνύουν την επικοινωνία μέσω υποδοχών ροής στο πεδίο Unix.*

```
/* File: un_str_server.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <sys/un.h>                                  /* For Unix sockets */
#include <stdio.h>                                          /* For I/O */
#include <stdlib.h>                                        /* For exit */
#include <string.h>                                 /* For strlen, bzero */
#include <signal.h>                                      /* For signals */

void reverse(char *);
void sigchld_handler(int);

main(int argc, char *argv[])         /* Server with Unix stream sockets */
{  int sock, newsock; char buf[256];
   unsigned int serverlen, clientlen;
   struct sockaddr_un server, client;
   struct sockaddr *serverptr, *clientptr;
   if (argc < 2) {                 /* Check if socket filename is given */
      printf("Please give the socket filename\n");
      exit(1); }
   signal(SIGCHLD, sigchld_handler); /* To be informed when child exits */
   if ((sock = socket(PF_UNIX, SOCK_STREAM, 0)) < 0) { /* Create socket */
      perror("socket"); exit(1); }
   server.sun_family = PF_UNIX;                          /* Unix domain */
   strcpy(server.sun_path, argv[1]);                  /* My Unix address */
   unlink(argv[1]);              /* Remove socket filename if it exists */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   if (bind(sock, serverptr, serverlen) < 0) {/* Bind socket to address */
      perror("bind"); exit(1); }
   if (listen(sock, 1) < 0) {                 /* Listen for connections */
      perror("listen"); exit(1); }
   printf("Listening for connections to socket %s\n", argv[1]);
```

```
   while(1) {
      clientptr = (struct sockaddr *) &client;
      clientlen = sizeof client;
      while ((newsock = accept(sock, clientptr, &clientlen)) < 0);
      printf("Accepted connection\n");             /* Accept connection */
      switch (fork()) {          /* Create child for serving the client */
         case -1:
            perror("fork"); exit(1);
         case 0:                                        /* Child process */
            do {
               bzero(buf, sizeof buf);             /* Initialize buffer */
               if (read(newsock, buf, sizeof buf) < 0) { /* Get message */
                  perror("read"); exit(1); }
               printf("Read string: %s\n", buf);
               reverse(buf);                          /* Reverse message */
               if (write(newsock, buf, sizeof buf) < 0){/* Send message */
                  perror("write"); exit(1); }
            } while (strcmp(buf, "dne") != 0);       /* Finish on "end" */
            close(newsock);                             /* Close socket */
            exit(0); } } }

void reverse(char *s)                /* Function for reversing a string */
{  int c, i, j;
   for (i = 0, j = strlen(s) - 1 ; i < j ; i++, j--) {
      c = s[i];
      s[i] = s[j];
      s[j] = c; } }

void sigchld_handler(sig)                          /* Handler for SIGCHLD */
{  int status;
   wait(&status); }
```

```
/* File: un_str_client.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <sys/un.h>                                  /* For Unix sockets */
#include <stdio.h>                                          /* For I/O */
#include <stdlib.h>                                        /* For exit */
#include <string.h>                                 /* For strlen, bzero */
main(int argc, char *argv[])         /* Client with Unix stream sockets */
{  int sock; char buf[256]; unsigned int serverlen;
   struct sockaddr_un server;
   struct sockaddr *serverptr;
   if (argc < 2) {                 /* Check if socket filename is given */
      printf("Please give the socket filename\n"); exit(1); }
   if ((sock = socket(PF_UNIX, SOCK_STREAM, 0)) < 0) { /* Create socket */
      perror("socket"); exit(1); }
   server.sun_family = PF_UNIX;                          /* Unix domain */
   strcpy(server.sun_path, argv[1]);           /* Server's Unix address */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   if (connect(sock, serverptr, serverlen) < 0) { /* Request connection */
      perror("connect"); exit(1); }
   printf("Requested connection to socket %s\n", argv[1]);
   do {
      bzero(buf, sizeof buf);                     /* Initialize buffer */
      printf("Give input string: ");
      fgets(buf, sizeof buf, stdin);         /* Read message from stdin */
      buf[strlen(buf)-1] = '\0';            /* Remove newline character */
      if (write(sock, buf, sizeof buf) < 0) {           /* Send message */
         perror("write"); exit(1); }
      bzero(buf, sizeof buf);                      /* Initialize buffer */
      if (read(sock, buf, sizeof buf) < 0) {         /* Receive message */
         perror("read"); exit(1); }
      printf("Read string:       %s\n", buf);
   } while (strcmp(buf, "dne") != 0);                /* Finish on "end" */
   close(sock);                                         /* Close socket */
   exit(0); }
```

```
$ ./un_str_server str_socket
Listening for connections to socket str_socket
Accepted connection
Read string: Testing stream sockets in the Unix domain...
Read string: abcdefghijklmnopqrstuvwxyz
Read string: >>>>>>><<<<<<
Read string: Is it OK?
Read string: Fine!
Read string: end
^C
$ rm str_socket
$
```

```
$ ./un_str_client str_socket
Requested connection to socket str_socket
Give input string: Testing stream sockets in the Unix domain...
Read string:       ...niamod xinU eht ni stekcos maerts gnitseT
Give input string: abcdefghijklmnopqrstuvwxyz
Read string:       zyxwvutsrqponmlkjihgfedcba
Give input string: >>>>>>><<<<<<
Read string:       <<<<<<>>>>>>
Give input string: Is it OK?
Read string:       ?KO ti sI
Give input string: Fine!
Read string:       !eniF
Give input string: end
Read string:       dne
$
```

- *Να γραφούν client και server προγράμματα C που να επιδεικνύουν την επικοινωνία μέσω τηλεγραφικών υποδοχών στο πεδίο Internet.*

```
/* File: int_dgr_server.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <netinet/in.h>                           /* For Internet sockets */
#include <netdb.h>                                   /* For gethostbyaddr */
#include <stdio.h>                                             /* For I/O */
#include <stdlib.h>                                           /* For exit */
#include <string.h>                                          /* For bzero */
main(int argc, char *argv[])   /* Server with Internet datagram sockets */
{  int n, port, sock; char buf[256]; unsigned int serverlen, clientlen;
   struct sockaddr_in server, client;
   struct sockaddr *serverptr, *clientptr; struct hostent *rem;
   if ((sock = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {  /* Create socket */
      perror("socket"); exit(1); }
   server.sin_family = PF_INET;                        /* Internet domain */
   server.sin_addr.s_addr = htonl(INADDR_ANY);   /* My Internet address */
   server.sin_port = htons(0);                         /* Select any port */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   if (bind(sock, serverptr, serverlen) < 0) {/* Bind socket to address */
      perror("bind"); exit(1); }
   if (getsockname(sock, serverptr, &serverlen) < 0) { /* Selected port */
      perror("getsockname"); exit(1); }
   printf("Socket port: %d\n", ntohs(server.sin_port));
   clientptr = (struct sockaddr *) &client;
   clientlen = sizeof client;
   while(1) {
      bzero(buf, sizeof buf);                       /* Initialize buffer */
      if ((n = recvfrom(sock, buf, sizeof buf, 0, clientptr,
                        &clientlen)) < 0) {
         perror("recvfrom"); exit(1); }              /* Receive message */
      if ((rem = gethostbyaddr((char *) &client.sin_addr.s_addr,
            sizeof client.sin_addr.s_addr, client.sin_family)) == NULL) {
         perror("gethostbyaddr"); exit(1); }   /* Find client's address */
      printf("Received from %s: %s\n", rem -> h_name, buf);
      if (sendto(sock, buf, n, 0, clientptr, clientlen) < 0) {
         perror("sendto"); exit(1); } } }              /* Send message */
```

```
/* File: int_dgr_client.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <netinet/in.h>                          /* For Internet sockets */
#include <netdb.h>                                 /* For gethostbyname */
#include <stdio.h>                                            /* For I/O */
#include <stdlib.h>                                          /* For exit */
#include <string.h>                             /* For strlen, bzero, bcopy */
main(int argc, char *argv[])   /* Client with Internet datagram sockets */
{  int port, sock; char buf[256];
   unsigned int serverlen, clientlen;
   struct sockaddr_in server, client;
   struct sockaddr *serverptr, *clientptr; struct hostent *rem;
   if (argc < 3) {     /* Are server's host name and port number given? */
      printf("Please give host name and port number\n"); exit(1); }
   if ((sock = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {  /* Create socket */
      perror("socket"); exit(1); }
   if ((rem = gethostbyname(argv[1])) == NULL) { /* Find server address */
      perror("gethostbyname"); exit(1); }
   port = atoi(argv[2]);              /* Convert port number to integer */
   server.sin_family = PF_INET;                        /* Internet domain */
   bcopy((char *) rem -> h_addr, (char *) &server.sin_addr,
         rem -> h_length);
   server.sin_port = htons(port); /* Server's Internet address and port */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   client.sin_family = PF_INET;                        /* Internet domain */
   client.sin_addr.s_addr = htonl(INADDR_ANY);   /* My Internet address */
   client.sin_port = htons(0);                         /* Select any port */
   clientptr = (struct sockaddr *) &client;
   clientlen = sizeof client;
   if (bind(sock, clientptr, clientlen) < 0) {/* Bind socket to address */
      perror("bind"); exit(1); }
   while (fgets(buf, sizeof buf, stdin) != NULL) { /* Read continuously
                                                     messages from stdin */
      buf[strlen(buf)-1] = '\0';            /* Remove newline character */
      if (sendto(sock, buf, strlen(buf)+1, 0, serverptr, serverlen) < 0) {
         perror("sendto"); exit(1); }                   /* Send message */
      bzero(buf, sizeof buf);                       /* Initialize buffer */
      if (recvfrom(sock, buf, sizeof buf, 0, serverptr, &serverlen) < 0) {
         perror("recvfrom"); exit(1); }              /* Receive message */
      printf("%s\n", buf); } }
```

```
$ ./int_dgr_server
Socket port: 32772
Received from galini.di.uoa.gr: prompt
Received from galini.di.uoa.gr: timeout=50
Received from galini.di.uoa.gr: default=linux
Received from galini.di.uoa.gr: boot=/dev/hda
Received from galini.di.uoa.gr: map=/boot/map
Received from galini.di.uoa.gr: install=/boot/boot.b
Received from galini.di.uoa.gr: message=/boot/message
Received from galini.di.uoa.gr: linear
Received from galini.di.uoa.gr:
Received from galini.di.uoa.gr: image=/boot/vmlinuz-2.4.7-10
Received from galini.di.uoa.gr:         label=linux
Received from galini.di.uoa.gr:         read-only
Received from galini.di.uoa.gr:         root=/dev/hda5
Received from galini.di.uoa.gr:
Received from galini.di.uoa.gr: other=/dev/hda1
Received from galini.di.uoa.gr:         optional
Received from galini.di.uoa.gr:         label=DOS
^C
$
```

```
$ ./int_dgr_client knossos.di.uoa.gr 32772 < /etc/lilo.conf
prompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
linear

image=/boot/vmlinuz-2.4.7-10
        label=linux
        read-only
        root=/dev/hda5

other=/dev/hda1
        optional
        label=DOS
$
```

- *Να γραφούν client και server προγράμματα C που να επιδεικνύουν την επικοινωνία μέσω τηλεγραφικών υποδοχών στο πεδίο Unix.*

```
/* File: un_dgr_server.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <sys/un.h>                                  /* For Unix sockets */
#include <stdio.h>                                         /* For I/O */
#include <stdlib.h>                                       /* For exit */
#include <string.h>                                      /* For bzero */
main(int argc, char *argv[])       /* Server with Unix datagram sockets */
{  int n, sock; char buf[256];
   unsigned int serverlen, clientlen;
   struct sockaddr_un server, client;
   struct sockaddr *serverptr, *clientptr;
   if (argc < 2) {                 /* Check if socket filename is given */
      printf("Please give the socket filename\n"); exit(1); }
   if ((sock = socket(PF_UNIX, SOCK_DGRAM, 0)) < 0) {  /* Create socket */
      perror("socket"); exit(1); }
   server.sun_family = PF_UNIX;                          /* Unix domain */
   strcpy(server.sun_path, argv[1]);                /* My Unix address */
   unlink(argv[1]);              /* Remove socket filename if it exists */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   if (bind(sock, serverptr, serverlen) < 0) {/* Bind socket to address */
      perror("bind"); exit(1); }
   printf("Waiting for data to ping\n");
   clientptr = (struct sockaddr *) &client;
   clientlen = sizeof client;
   while(1) {
      bzero(buf, sizeof buf);                     /* Initialize buffer */
      if ((n = recvfrom(sock, buf, sizeof buf, 0, clientptr,
                        &clientlen)) < 0) {
         perror("recvfrom"); exit(1); }              /* Receive message */
      printf("Received from %s: %s\n", client.sun_path, buf);
      if (sendto(sock, buf, n, 0, clientptr, clientlen) < 0) {
         perror("sendto"); exit(1); } } }              /* Send message */
```

```
/* File: un_dgr_client.c */
#include <sys/types.h>                                  /* For sockets */
#include <sys/socket.h>                                 /* For sockets */
#include <sys/un.h>                                  /* For Unix sockets */
#include <stdio.h>                                          /* For I/O */
#include <stdlib.h>                                        /* For exit */
#include <string.h>                                 /* For strlen, bzero */
main(int argc, char *argv[])       /* Client with Unix datagram sockets */
{  int sock; char buf[256], sname[20];
   unsigned int serverlen, clientlen;
   struct sockaddr_un server, client;
   struct sockaddr *serverptr, *clientptr;
   if (argc < 2) {                 /* Check if socket filename is given */
      printf("Please give the socket filename\n"); exit(1); }
   if ((sock = socket(PF_UNIX, SOCK_DGRAM, 0)) < 0) {  /* Create socket */
      perror("socket"); exit(1); }
   server.sun_family = PF_UNIX;                             /* Unix domain */
   strcpy(server.sun_path, argv[1]);           /* Server's Unix address */
   serverptr = (struct sockaddr *) &server;
   serverlen = sizeof server;
   sprintf(sname, "s%d", getpid()); /* Create my socket filename (=PID) */
   client.sun_family = PF_UNIX;                             /* Unix domain */
   strcpy(client.sun_path, sname);                    /* My Unix address */
   clientptr = (struct sockaddr *) &client;
   clientlen = sizeof client;
   if (bind(sock, clientptr, clientlen) < 0) {/* Bind socket to address */
      perror("bind"); exit(1); }
   while (fgets(buf, sizeof buf, stdin) != NULL) { /* Read continuously
                                                    messages from stdin */
      buf[strlen(buf)-1] = '\0';             /* Remove newline character */
      if (sendto(sock, buf, strlen(buf)+1, 0, serverptr, serverlen) < 0) {
         perror("sendto"); exit(1); }                    /* Send message */
      bzero(buf, sizeof buf);                         /* Initialize buffer */
      if (recvfrom(sock, buf, sizeof buf, 0, serverptr, &serverlen) < 0) {
         perror("recvfrom"); exit(1); }               /* Receive message */
      printf("%s\n", buf); }
   unlink(sname); }                          /* Remove my socket filename */
```

```
$ ./un_dgr_server dgr_socket
Waiting for data to ping
Received from s15386: root:x:0:root
Received from s15386: bin:x:1:root,bin,daemon
Received from s15386: daemon:x:2:root,bin,daemon
Received from s15386: sys:x:3:root,bin,adm
Received from s15386: adm:x:4:root,adm,daemon
Received from s15386: tty:x:5:
Received from s15386: disk:x:6:root
Received from s15386: lp:x:7:daemon,lp
Received from s15386: mem:x:8:
Received from s15386: kmem:x:9:
^C
$ rm dgr_socket
$
```

```
$ head /etc/group | ./un_dgr_client dgr_socket
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
$
```

Επικοινωνία μεταξύ διεργασιών *a la* System V

- Πραγματικά ασύγχρονη επικοινωνία στον ίδιο υπολογιστή
- Καθολικά προσπελάσιμοι πόροι μέσω του πυρήνα του Unix
  - Ουρές μηνυμάτων
  - Κοινή μνήμη
  - Σηματοφόροι
- Ταυτοποίηση πόρων μέσω κλειδιών (τύπου `key_t`) και πρόσβαση στις εσωτερικές δομές τους (`struct msqid_ds`, `struct shmid_ds` και `struct semid_ds`) μέσω προσδιοριστών που επιστρέφονται και χρησιμοποιούνται από κατάλληλες κλήσεις συστήματος (οι οποίες, ως συνήθως, επιστρέφουν -1 σε περίπτωση λάθους)
- Δικαιώματα προστασίας πόρων (read/write) που καταχωρούνται σε δομές `struct ipc_perm`
- Μοναδικά κλειδιά (τύπου `key_t`) μπορούν να παραχθούν καλώντας τη συνάρτηση

  ```
  key_t ftok(const char *pathname, int proj_id)
  ```

  δίνοντάς της ένα όνομα-μονοπάτι `pathname` και έναν ακέραιο `proj_id`
- Απαιτήσεις
  - `#include <sys/types.h>`
  - `#include <sys/ipc.h>`

Ουρές μηνυμάτων

- Χρησιμεύουν στην ανταλλαγή μηνυμάτων μεταξύ διεργασιών
- Η αποστέλλουσα διεργασία επισυνάπτει έναν τύπο στο μήνυμα που στέλνει και η παραλαμβάνουσα διεργασία μπορεί να ζητήσει την παραλαβή μηνύματος συγκεκριμένου τύπου
- Πριν το σώμα ενός μηνύματος πρέπει να προηγείται ο τύπος του (`long`) σε bytes, δηλαδή στις κλήσεις συστήματος για αποστολή και παραλαβή μηνυμάτων χρειάζεται να δίνεται ένας δείκτης σε δομή σαν την

```
struct message {
   long mtype;
   char mtext[MSGSIZE]; };
```

- Απαίτηση: `#include <sys/msg.h>`

- Κλήση συστήματος `msgget`
  - `int msgget(key_t key, int msgflag)`
  - Επιστρέφει έναν προσδιοριστή για την ουρά μηνυμάτων που αντιστοιχεί στο κλειδί `key`
  - Το `msgflag` είναι ένας ακέραιος όπου τίθενται τα επιθυμητά δικαιώματα προστασίας της ουράς μηνυμάτων, καθώς επίσης και πρόσθετες απαιτήσεις (υπό τη μορφή διάζευξης συμβολικών ονομάτων) σχετικές με τη δημιουργία της ουράς μηνυμάτων, όπως:

    `IPC_CREAT`: Αν δεν υπάρχει πόρος (ουρά μηνυμάτων) που αντιστοιχεί στο `key` να δημιουργηθεί νέος (αντί να επιστραφεί λάθος), ενώ αν υπάρχει πόρος, να προσπελασθεί αυτός

    `IPC_EXCL`: Σε συνδυασμό με το προηγούμενο, αν δεν υπάρχει πόρος να δημιουργηθεί, αλλά αν υπάρχει να επιστραφεί λάθος
  - Περιπτωσιολογία για το `msgflag`

	`PERMS`	`PERMS|IPC_CREAT`	`PERMS|IPC_CREAT|IPC_EXCL`
υπάρχει πόρος	χρήση του πόρου	χρήση του πόρου	λάθος
δεν υπάρχει πόρος	λάθος	δημιουργία και χρήση νέου πόρου	δημιουργία και χρήση νέου πόρου

- Κλήσεις συστήματος `msgrcv` και `msgsnd`
  - `int msgrcv(int msqid, void *ptr, int len, long type, int flag)`
  - `int msgsnd(int msqid, void *ptr, int len, int flag)`
  - Με την `msgrcv` παραλαμβάνεται ένα μήνυμα τύπου `type`, επιστρέφοντας το μέγεθος του μηνύματος, από την ουρά μηνυμάτων με προσδιοριστή `msqid` (αν το `type` είναι 0, παραλαμβάνεται το πρώτο μήνυμα, ανεξαρτήτως τύπου) και με την `msgsnd` αποστέλλεται ένα μήνυμα στην ουρά
  - Ο τύπος και το σώμα του μηνύματος βρίσκονται σε μία δομή στην οποία δείχνει το `ptr` και το `len` είναι το μέγεθος του σώματος του μηνύματος για την `msgsnd` ή το μέγιστο του μεγέθους αυτού για την `msgrcv`
  - Στο `flag` τίθεται το 0, εκτός ειδικών περιπτώσεων
- Κλήση συστήματος `msgctl`
  - `int msgctl(int msqid, int cmd, struct msqid_ds *buff)`
  - Εκτελεί την ενέργεια `cmd` επάνω στην ουρά μηνυμάτων που αντιστοιχεί στον προσδιοριστή `msqid`
  - Μία ενέργεια είναι η `IPC_STAT` με την οποία συμπληρώνονται τα πεδία της δομής `*buff` με τα χαρακτηριστικά της ουράς μηνυμάτων που ενδιαφέρει
  - Η συνηθέστερη ενέργεια είναι η `IPC_RMID` που καταστρέφει την ουρά μηνυμάτων (στο `buff` αρκεί να τεθεί 0 αφού προσαρμοσθεί σε `(struct msqid_ds *)`)

- *Να γραφούν client και server προγράμματα C που να επιδεικνύουν την επικοινωνία μεταξύ τους μέσω ουρών μηνυμάτων.*

```
/* File: msgq_server.c */
#include <sys/types.h>                              /* For System V IPC */
#include <sys/ipc.h>                                /* For System V IPC */
#include <sys/msg.h>                               /* For message queues */
#include <stdio.h>                                          /* For I/O */
#include <stdlib.h>                                        /* For exit */
#include <string.h>                               /* For strcpy, strlen */
#define MSGSIZE 256          /* Maximum size of message to communicate */
#define KEY (key_t)5678                /* Key value of message queue */
#define PERMS 0666                    /* Permissions of message queue */
#define SERVER_MTYPE 27L             /* Type of message sent by server */
#define CLIENT_MTYPE 42L             /* Type of message sent by client */

struct message {                                  /* Message structure */
   long mtype;
   char mtext[MSGSIZE]; };

main()
{  int qid;
   struct message sbuf, rbuf;
   if ((qid = msgget(KEY, PERMS | IPC_CREAT)) < 0) {
      perror("msgget"); exit(1); }              /* Create message queue */
   printf("Created message queue with identifier %d\n", qid);
   sbuf.mtype = SERVER_MTYPE;                  /* Server's message type */
   strcpy(sbuf.mtext, "A message from server");     /* Server's message */
   if (msgsnd(qid, &sbuf, strlen(sbuf.mtext)+1, 0) < 0) {
      perror("msgsnd"); exit(1); }                       /* Send message */
   printf("Sent message:     %s\n", sbuf.mtext);
   if (msgrcv(qid, &rbuf, MSGSIZE, CLIENT_MTYPE, 0) < 0) {
      perror("msgrcv"); exit(1); }                   /* Receive message */
   printf("Received message: %s\n", rbuf.mtext);
   if (msgctl(qid, IPC_RMID, (struct msqid_ds *) 0) < 0) {
      perror("msgctl"); exit(1); }             /* Destroy message queue */
   printf("Removed message queue with identifier %d\n", qid); }
```

```
/* File: msgq_client.c */
#include <sys/types.h>                              /* For System V IPC */
#include <sys/ipc.h>                                /* For System V IPC */
#include <sys/msg.h>                               /* For message queues */
#include <stdio.h>                                          /* For I/O */
#include <stdlib.h>                                        /* For exit */
#include <string.h>                                /* For strcpy, strlen */
#define MSGSIZE 256           /* Maximum size of message to communicate */
#define KEY (key_t)5678                  /* Key value of message queue */
#define PERMS 0666                      /* Permissions of message queue */
#define SERVER_MTYPE 27L              /* Type of message sent by server */
#define CLIENT_MTYPE 42L              /* Type of message sent by client */

struct message {                                  /* Message structure */
   long mtype;
   char mtext[MSGSIZE]; };

main()
{  int qid;
   struct message sbuf, rbuf;
   if ((qid = msgget(KEY, PERMS)) < 0) {
      perror("msgget"); exit(1); }              /* Access message queue */
   printf("Accessing message queue with identifier %d\n", qid);
   if (msgrcv(qid, &rbuf, MSGSIZE, SERVER_MTYPE, 0) < 0) {
      perror("msgrcv"); exit(1); }                  /* Receive message */
   printf("Received message: %s\n", rbuf.mtext);
   sbuf.mtype = CLIENT_MTYPE;                   /* Client's message type */
   strcpy(sbuf.mtext, "A message from client");     /* Client's message */
   if (msgsnd(qid, &sbuf, strlen(sbuf.mtext)+1, 0) < 0) {
      perror("msgsnd"); exit(1); }                      /* Send message */
   printf("Sent message:     %s\n", sbuf.mtext); }
```

```
$ ./msgq_server
Created message queue with identifier 98304
Sent message:     A message from server
Received message: A message from client
Removed message queue with identifier 98304
$
```

```
$ ./msgq_client
Accessing message queue with identifier 98304
Received message: A message from server
Sent message:     A message from client
$
```

Κοινή μνήμη

- Δυνατότητα επικοινωνίας μεταξύ διεργασιών μέσω καταχώρησης και ανάγνωσης πληροφοριών σε περιοχή μνήμης που είναι προσπελάσιμη από όλες τις διεργασίες
- Ανάγκη συγχρονισμού των διεργασιών, συνήθως μέσω σηματοφόρων
- Απαίτηση: `#include <sys/shm.h>`
- Κλήση συστήματος `shmget`
  - `int shmget(key_t key, int size, int shmflag)`
  - Επιστρέφει έναν προσδιοριστή για την κοινή μνήμη μεγέθους `size` που αντιστοιχεί στο κλειδί `key`
  - Στο `shmflag` τίθενται τα επιθυμητά δικαιώματα προστασίας καθώς και πρόσθετες απαιτήσεις (`IPC_CREAT` και `IPC_EXCL`, με την ίδια σημασία που έχουν και στις ουρές μηνυμάτων) σχετικές με τη δημιουργία της κοινής μνήμης

- Κλήση συστήματος `shmat`
  - `char *shmat(int shmid, char *addr, int flag)`
  - Προσαρτά την κοινή μνήμη που αντιστοιχεί στον προσδιοριστή `shmid` στην περιοχή μνήμης που έχει πρόσβαση η καλούσα διεργασία και επιστρέφει την κατάλληλη διεύθυνση
  - Μέσω των `addr` και `flag` μπορεί να ζητηθεί η προσάρτηση σε συγκεκριμένη περιοχή μνήμης, αλλά η συνηθισμένη χρήση της `shmat` είναι να αφεθεί η επιλογή αυτή στον πυρήνα θέτοντας 0 στα `addr` και `flag` (το πρώτο προσαρμοσμένο σε `(char *)`)
- Κλήση συστήματος `shmdt`
  - `int shmdt(char *addr)`
  - Αποσπά την προσαρτημένη στο `addr` κοινή μνήμη
- Κλήση συστήματος `shmctl`
  - `int shmctl(int shmid, int cmd,`
    `struct shmid_ds *buff)`
  - Εκτελεί την ενέργεια `cmd` στην κοινή μνήμη που αντιστοιχεί στον προσδιοριστή `shmid`
  - Αντίστοιχα με τις δυνατότητες που υπάρχουν για τις ουρές μηνυμάτων (μέσω της `msgctl`), με την ενέργεια `IPC_STAT` συμπληρώνονται τα πεδία της δομής `*buff` με τα χαρακτηριστικά της κοινής μνήμης, ενώ η συνηθέστερη ενέργεια είναι η `IPC_RMID` που καταστρέφει την κοινή μνήμη (στο `buff` αρκεί να τεθεί 0 αφού προσαρμοσθεί σε `(struct shmid_ds *)`)

Σηματοφόροι

- Μηχανισμός συγχρονισμού διεργασιών για την αποκλειστική διαχείριση κοινών πόρων (π.χ. κοινής μνήμης)
- Πριν την είσοδο σε κρίσιμο τμήμα του προγράμματός της, μία διεργασία ζητά την απαιτούμενη άδεια από έναν ελεγκτή σηματοφόρο (αναμένοντας, αν χρειάζεται, μέχρι να της δοθεί, οπότε δεσμεύει το απαιτούμενο μέρος του πόρου που ελέγχει ο σηματοφόρος) και μετά την έξοδο από το κρίσιμο τμήμα αποδεσμεύει το δεσμευμένο μέρος του πόρου
- Η δέσμευση γίνεται με κατάλληλη μείωση (DOWN ή P λειτουργία κατά Dijkstra) της τιμής του σηματοφόρου, εφ΄ όσον μετά τη μείωση η νέα τιμή θα είναι ≥ 0, και η αποδέσμευση με κατάλληλη αύξηση (UP ή V λειτουργία κατά Dijkstra) της τιμής του σηματοφόρου

- Με τις κλήσεις συστήματος που θα ακολουθήσουν δημιουργούνται και χρησιμοποιούνται σύνολα από σηματοφόρους και ο πυρήνας εγγυάται ότι ένα σύνολο λειτουργιών επάνω σε ένα τέτοιο σύνολο σηματοφόρων είναι ατομική διαδικασία
- Απαίτηση: `#include <sys/sem.h>`
- Κλήση συστήματος `semget`
  - `int semget(key_t key, int nsems, int semflag)`
  - Επιστρέφει έναν προσδιοριστή για ένα σύνολο από `nsems` σηματοφόρους που αντιστοιχεί στο κλειδί `key`
  - Στο `semflag` τίθενται τα επιθυμητά δικαιώματα προστασίας καθώς και πρόσθετες απαιτήσεις (`IPC_CREAT` και `IPC_EXCL`, με την ίδια σημασία που έχουν στις ουρές μηνυμάτων και στην κοινή μνήμη) σχετικές με τη δημιουργία του συνόλου σηματοφόρων

- Κλήση συστήματος semop
  - `int semop(int semid, struct sembuf *opstr, int nops)`
  - Εκτελεί στο σύνολο σηματοφόρων που προσδιορίζονται από το `semid` τις λειτουργίες που καθορίζονται σε ένα πίνακα μεγέθους `nops` από δομές `struct sembuf` το πρώτο στοιχείο του οποίου δείχνει το `opstr`
  - Η δομή `struct sembuf` ορίζεται σαν

```
struct sembuf {
    short sem_num;
    short sem_op;
    short sem_flg; };
```

    και περιγράφει τη λειτουργία μεταβολής της τιμής του υπ΄ αριθμόν `sem_num` σηματοφόρου του συνόλου (από 0 έως `nsems-1`) κατά `sem_op` (<0 για δέσμευση και >0 για αποδέσμευση), ενώ το `sem_flg` τίθεται συνήθως 0, εκτός ειδικών περιπτώσεων

- Κλήση συστήματος `semctl`
  - `int semctl(int semid, int semnum, int cmd, union semun arg)`
  - Εκτελεί την ενέργεια `cmd` στον `semnum` σηματοφόρο του συνόλου σηματοφόρων (ή, ανάλογα με την ενέργεια, σε ολόκληρο το σύνολο) που αντιστοιχεί στον προσδιοριστή `semid`
  - Η ένωση `union semun` είναι ορισμένη σαν

```
union semun {
    int              val;
    struct semid_ds *buff;
    unsigned short  *array; };
```

    και χρησιμεύει για να καλύψει όλες τις δυνατότητες που υπάρχουν για έλεγχο συνόλων σηματοφόρων
  - Με την ενέργεια `IPC_STAT` συμπληρώνονται τα πεδία της δομής `*(arg.buff)` με τα χαρακτηριστικά του συνόλου σηματοφόρων
  - Με την ενέργεια `SETVAL` τίθεται σαν τιμή ενός σηματοφόρου το `arg.val` και με την ενέργεια `GETVAL` επιστρέφει η `semctl` την τιμή του σηματοφόρου
  - Με τις ενέργειες `SETALL` και `GETALL` τίθενται τιμές στους σηματοφόρους του συνόλου (από τον πίνακα στην αρχή του οποίου δείχνει το `arg.array`) ή επιστρέφονται οι τιμές των σηματοφόρων (στον πίνακα `arg.array`)
  - Με την ενέργεια `IPC_RMID` καταστρέφεται το σύνολο των σηματοφόρων

- *Να γραφούν client και server προγράμματα C που να επιδεικνύουν την επικοινωνία μεταξύ τους μέσω κοινής μνήμης και το συγχρονισμό τους μέσω σηματοφόρων.*

```
/* File: shm_sem_server.c */
#include <sys/types.h>                                  /* For System V IPC */
#include <sys/ipc.h>                                    /* For System V IPC */
#include <sys/shm.h>                                   /* For shared memory */
#include <sys/sem.h>                                      /* For semaphores */
#include <stdio.h>                                               /* For I/O */
#include <stdlib.h>                                             /* For exit */
#include <string.h>                                  /* For strcpy, strlen */
#define SHMKEY (key_t)4321                 /* Key value of shared memory */
#define SEMKEY (key_t)9876                 /* Key value of semaphore set */
#define SHMSIZ 256                              /* Size of shared memory */
#define PERMS 0600    /* Permissions of shared memory and semaphore set */
union semun {                                       /* Union for semaphores */
   int val; struct semid_ds *buff; unsigned short *array; };
main()
{  int shmid, semid; char line[128], *shmem;
   struct sembuf oper[1] = {0, 1, 0}; union semun arg;
   if ((shmid = shmget(SHMKEY, SHMSIZ, PERMS | IPC_CREAT)) < 0) {
      perror("shmget"); exit(1); }              /* Create shared memory */
   printf("Created shared memory region with identifier %d\n", shmid);
   if ((semid = semget(SEMKEY, 1, PERMS | IPC_CREAT)) < 0) {
      perror("semget"); exit(1); }  /* Create semaphore set with 1 item */
   printf("Created semaphore with identifier %d\n", semid);
   arg.val=0;
   if (semctl(semid, 0, SETVAL, arg) < 0) {
      perror("semctl"); exit(1); }  /* Initialize semaphore for locking */
   printf("Initialized semaphore to lock shared memory region\n");
   if ((shmem = shmat(shmid, (char *) 0, 0)) == (char *) -1) {
      perror("shmat"); exit(1); }/* Attach shared memory region locally */
   printf("Attached shared memory region\nGive input line: ");
   fgets(line, sizeof line, stdin); line[strlen(line)-1] = '\0';
   strcpy(shmem, line);               /* Write message to shared memory */
   printf("Wrote to shared memory region: %s\n", line);
   if (semop(semid, &oper[0], 1) < 0) {
      perror("semop"); exit(1); }       /* Make shared memory available */
   shmdt(shmem);                          /* Detach shared memory region */
   printf("Released shared memory region\n"); }
```

```
/* File: shm_sem_client.c */
#include <sys/types.h>                                  /* For System V IPC */
#include <sys/ipc.h>                                    /* For System V IPC */
#include <sys/shm.h>                                   /* For shared memory */
#include <sys/sem.h>                                      /* For semaphores */
#include <stdio.h>                                               /* For I/O */
#include <stdlib.h>                                             /* For exit */
#define SHMKEY (key_t)4321                 /* Key value of shared memory */
#define SEMKEY (key_t)9876                 /* Key value of semaphore set */
#define SHMSIZ 256                              /* Size of shared memory */
#define PERMS 0600    /* Permissions of shared memory and semaphore set */
main()
{  int shmid, semid;
   char *shmem;
   struct sembuf oper[1] = {0, -1, 0};
   if ((shmid = shmget(SHMKEY, SHMSIZ, PERMS)) < 0) {
      perror("shmget"); exit(1); }               /* Access shared memory */
   printf("Accessing shared memory region with identifier %d\n", shmid);
   if ((semid = semget(SEMKEY, 1, PERMS)) < 0) {
      perror("semget"); exit(1); }               /* Access semaphore set */
   printf("Accessing semaphore with identifier %d\n", semid);
   if ((shmem = shmat(shmid, (char *) 0, 0)) == (char *) -1) {
      perror("shmat"); exit(1); }/* Attach shared memory region locally */
   printf("Attached shared memory region\n");
   printf("Asking for access to shared memory region\n");
   if (semop(semid, &oper[0], 1) < 0) {
      perror("semop"); exit(1); }/* Ask if you may access shared memory */
   printf("Read from shared memory region: %s\n", shmem);  /* Accessing */
   shmdt(shmem);                            /* Detach shared memory region */
   if (shmctl(shmid, IPC_RMID, (struct shmid_ds *) 0) < 0) {
      perror("shmctl"); exit(1); }             /* Destroy shared memory */
   printf("Removed shared memory region with identifier %d\n", shmid);
   if (semctl(semid, 0, IPC_RMID, 0) < 0) {
      perror("semctl"); exit(1); }             /* Destroy semaphore set */
   printf("Removed semaphore with identifier %d\n", semid); }
```

```
$ ./shm_sem_server
Created shared memory region with identifier 163840
Created semaphore with identifier 131072
Initialized semaphore to lock shared memory region
Attached shared memory region
Give input line: To be saved in shared memory
Wrote to shared memory region: To be saved in shared memory
Released shared memory region
$
```

```
$ ./shm_sem_client
Accessing shared memory region with identifier 163840
Accessing semaphore with identifier 131072
Attached shared memory region
Asking for access to shared memory region
Read from shared memory region: To be saved in shared memory
Removed shared memory region with identifier 163840
Removed semaphore with identifier 131072
$
```

Νήματα

- Ένα ή περισσότερα νήματα μπορούν να εκτελούνται στο πλαίσιο μίας διεργασίας
- Η βασική μονάδα που χρονοδρομολογείται από το λειτουργικό σύστημα είναι το νήμα και όχι η διεργασία
- Τα νήματα των διεργασιών εκτελούνται ψευδο-παράλληλα, αλλά μπορούν να εκτελεσθούν και πραγματικά παράλληλα σε συστήματα με πολλούς επεξεργαστές
- Οποιοδήποτε νήμα μίας διεργασίας μπορεί να δημιουργήσει άλλα νήματα
- Όλα τα νήματα μίας διεργασίας μοιράζονται τον ίδιο χώρο διευθύνσεων, καθώς και άλλα χαρακτηριστικά της διεργασίας (π.χ. κώδικας, περιγραφείς αρχείων κ.λ.π.) αλλά το καθένα έχει τη δική του στοίβα εκτέλεσης και δείκτη προγράμματος
- Το λειτουργικό σύστημα μπορεί να εκτελέσει πολύ γρηγορότερα την εναλλαγή νημάτων σε σχέση με την εναλλαγή διεργασιών
- ANSI/IEEE POSIX 1003.1c – 1995 standard

- Όλες οι συναρτήσεις βιβλιοθήκης που ακολουθούν και χρησιμοποιούνται για διαχείριση νημάτων απαιτούν

  ```
  #include <pthread.h>
  ```

  Τα προγράμματα στα οποία χρησιμοποιούνται οι συναρτήσεις αυτές πρέπει να μεταγλωττίζονται με την εντολή

  ```
  cc -o <filename> <filename>.c -lpthread
  ```

  έτσι ώστε στο δημιουργούμενο εκτελέσιμο να "φορτώνεται" και η βιβλιοθήκη για τα νήματα

- Οι συναρτήσεις βιβλιοθήκης για διαχείριση νημάτων δεν θέτουν τιμή, σε περίπτωση λάθους, στην εξωτερική μεταβλητή `errno`, συνεπώς δεν μπορεί να χρησιμοποιηθεί η συνάρτηση βιβλιοθήκης `perror` για την εκτύπωση κατάλληλου διαγνωστικού μηνύματος

- Σε περίπτωση λάθους κατά την κλήση κάποιας συνάρτησης διαχείρισης νημάτων, μπορεί να χρησιμοποιηθεί η συνάρτηση βιβλιοθήκης `strerror`, για την εκτύπωση κατάλληλου διαγνωστικού μηνύματος που αντιστοιχεί στον κωδικό του λάθους που επέστρεψε η συνάρτηση για το νήμα

- Συνάρτηση βιβλιοθήκης `strerror`
  - `char *strerror(int errnum)`
  - Επιστρέφει μία συμβολοσειρά που περιγράφει το λάθος το οποίο αντιστοιχεί στον κωδικό λάθους `errnum`
  - Απαίτηση: `#include <string.h>`

Δημιουργία και τερματισμός νημάτων

- Συνάρτηση βιβλιοθήκης `pthread_create`
  - `int pthread_create(pthread_t *thread, pthread_attr_t *attr, void *(*start)(void *), void *arg)`
  - Δημιουργεί ένα καινούργιο νήμα το οποίο εκτελεί τη συνάρτηση με διεύθυνση `start` και παράμετρο το `arg`
  - Η ταυτότητα του νήματος επιστρέφεται στο `*thread`
  - Μέσω του `attr` μπορούμε να ορίσουμε κάποια χαρακτηριστικά για το νήμα, αλλά συνήθως αφήνουμε τα default, δίνοντάς του την τιμή `NULL`
  - Επιστρέφει 0 σε επιτυχία
- Συνάρτηση βιβλιοθήκης `pthread_exit`
  - `void pthread_exit(void *retval)`
  - Τερματίζει το νήμα που την καλεί
  - Εφ΄ όσον το νήμα είναι συνενώσιμο, που είναι η default περίπτωση, ο κωδικός εξόδου `retval` είναι διαθέσιμος σε οποιοδήποτε άλλο νήμα της διεργασίας που περιμένει, μέσω της συνάρτησης `pthread_join` η οποία θα περιγραφεί στη συνέχεια, τον τερματισμό του συγκεκριμένου νήματος

- Συνάρτηση βιβλιοθήκης `pthread_join`
  - `int pthread_join(pthread_t thread, void **retaddr)`
  - Περιμένει τον τερματισμό του συνενώσιμου νήματος με ταυτότητα `thread`
  - Ο κωδικός εξόδου του νήματος που τερμάτισε, όπως δόθηκε με την `pthread_exit`, επιστρέφεται στο `*retaddr`
  - Επιστρέφει 0 σε επιτυχία
- Συνάρτηση βιβλιοθήκης `pthread_self`
  - `pthread_t pthread_self()`
  - Επιστρέφει την ταυτότητα του νήματος που την καλεί
- Συνάρτηση βιβλιοθήκης `pthread_detach`
  - `int pthread_detach(pthread_t thread)`
  - Μετατρέπει το νήμα με ταυτότητα `thread` από συνενώσιμο σε αποσπασμένο
  - Επιστρέφει 0 σε επιτυχία
  - Ένα αποσπασμένο νήμα ελευθερώνει αμέσως τους πόρους που έχει δεσμεύσει μόλις τερματίσει, ενώ ένα συνενώσιμο το κάνει αυτό μόνο όταν κάποιο άλλο νήμα ζητήσει να συνενωθεί μαζί του, μέσω της `pthread_join`
  - Η κλήση της `pthread_join` για ένα αποσπασμένο νήμα αποτυγχάνει

- Χρήση των συναρτήσεων pthread_create, pthread_exit, pthread_join και pthread_self

```
/* File: create_a_thread.c */
#include <stdio.h>                                  /* For printf */
#include <string.h>                                /* For strerror */
#include <stdlib.h>                                    /* For exit */
#include <pthread.h>                               /* For threads */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                             /* Error message printing function */
void *thread_f(void *);                  /* Forward declaration */

main()
{  pthread_t thr;
   int err, status;
   if (err = pthread_create(&thr, NULL, thread_f, NULL)) {
                                                    /* New thread */
      perror2("pthread_create", err);
      exit(1); }
   printf("I am original thread %d and I created thread %d\n",
          pthread_self(), thr);
   if (err = pthread_join(thr, (void **) &status)) {
                                             /* Wait for thread */
      perror2("pthread_join", err);              /* termination */
      exit(1); }
   printf("Thread %d exited with code %d\n", thr, status);
   pthread_exit(NULL); }

void *thread_f(void *argp)                    /* Thread function */
{  printf("I am the newly created thread %d\n", pthread_self());
   pthread_exit((void *) 47);
                          /* Not correct, according to Stefanos */
   /* See message http://www.di.uoa.gr/~syspro/getmsg.cgi?1404 */
}
```

```
$ ./create_a_thread
I am the newly created thread 1026
I am original thread 1024 and I created thread 1026
Thread 1026 exited with code 47
$
```

- Χρήση της συνάρτησης pthread_detach

```
/* File: detached_thread.c */
#include <stdio.h>                                  /* For printf */
#include <string.h>                                /* For strerror */
#include <stdlib.h>                                    /* For exit */
#include <pthread.h>                               /* For threads */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                             /* Error message printing function */
void *thread_f(void *);                  /* Forward declaration */

main()
{  pthread_t thr;
   int err, arg = 29;
   if (err = pthread_create(&thr, NULL, thread_f, (void *) &arg)) {
                                                    /* New thread */
      perror2("pthread_create", err);
      exit(1); }
   printf("I am original thread %d and I created thread %d\n",
          pthread_self(), thr);
   pthread_exit(NULL); }

void *thread_f(void *argp)                    /* Thread function */
{  int err;
   if (err = pthread_detach(pthread_self())) {/* Detach thread */
      perror2("pthread_detach", err);
      exit(1); }
   printf("I am thread %d and I was called with argument %d\n",
         pthread_self(), *(int *) argp);
   pthread_exit(NULL); }
```

```
$ ./detached_thread
I am original thread 1024 and I created thread 1026
I am thread 1026 and I was called with argument 29
$
```

- *Να γραφεί ένα πρόγραμμα C που να δημιουργεί ένα αριθμό από νήματα, καθένα από τα οποία να καθυστερεί να τερματίσει για ένα τυχαίο αριθμό δευτερολέπτων.*

```
/* File: random_sleeps.c */
#include <stdio.h>                                    /* For printf */
#include <string.h>                                  /* For strerror */
#include <stdlib.h>                      /* For srandom, random, exit */
#include <pthread.h>                                  /* For threads */
#define MAX_SLEEP 10              /* Maximum sleeping time in seconds */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                                 /* Error message printing function */
void *sleeping(void *);                         /* Forward declaration */

main(int argc, char *argv[])
{  int n, i, sl, err;
   pthread_t *tids;
   if (argc > 1) n = atoi(argv[1]);                  /* Make integer */
   else          exit(0);
   if (n > 50) {                              /* Avoid too many threads */
      printf("Number of threads should be up to 50\n"); exit(0); }
   if ((tids = malloc(n * sizeof(pthread_t))) == NULL) {
      perror("malloc"); exit(1); }
   srandom((unsigned int) time(NULL));       /* Initialize generator */
   for (i=0 ; i<n ; i++) {
      sl = random() % MAX_SLEEP + 1;  /* Sleeping time 1..MAX_SLEEP */
      if (err = pthread_create(tids+i, NULL, sleeping, (void *) sl)) {
                                                  /* Create a thread */
         perror2("pthread_create", err); exit(1); } }
   for (i=0 ; i<n ; i++)
      if (err = pthread_join(*(tids+i), NULL)) {
                                      /* Wait for thread termination */
         perror2("pthread_join", err); exit(1); }
   printf("all %d threads have terminated\n", n); }

void *sleeping(void *arg)
{  int sl = (int) arg;
   printf("thread %d sleeping %d seconds ...\n", pthread_self(), sl);
   sleep(sl);                              /* Sleep a number of seconds */
   printf("thread %d awakening\n", pthread_self());
   pthread_exit(NULL); }
```

```
$ ./random_sleeps 10
thread 1026 sleeping 8 seconds ...
thread 2051 sleeping 5 seconds ...
thread 3076 sleeping 1 seconds ...
thread 4101 sleeping 9 seconds ...
thread 5126 sleeping 6 seconds ...
thread 6151 sleeping 9 seconds ...
thread 7176 sleeping 2 seconds ...
thread 8201 sleeping 3 seconds ...
thread 9226 sleeping 3 seconds ...
thread 10251 sleeping 1 seconds ...
thread 3076 awakening
thread 10251 awakening
thread 7176 awakening
thread 8201 awakening
thread 9226 awakening
thread 2051 awakening
thread 5126 awakening
thread 1026 awakening
thread 6151 awakening
thread 4101 awakening
all 10 threads have terminated
$
```

Συγχρονισμός νημάτων με δυαδικούς σηματοφόρους

- Για το συγχρονισμό μεταξύ νημάτων που προτίθενται να προσπελάσουν κοινούς πόρους, η βιβλιοθήκη των POSIX νημάτων παρέχει μία απλοποιημένη εκδοχή σηματοφόρων, τους δυαδικούς σηματοφόρους (mutexes)
- Ένας δυαδικός σηματοφόρος μπορεί να βρίσκεται σε μία από δύο πιθανές καταστάσεις, να είναι κλειδωμένος ή ξεκλείδωτος
- Συνάρτηση βιβλιοθήκης `pthread_mutex_init`
  - `int pthread_mutex_init(pthread_mutex_t *mutex, pthread_mutexattr_t *attr)`
  - Αρχικοποιεί δυναμικά το σηματοφόρο `*mutex`
  - Μέσω του `attr` μπορούμε να ορίσουμε κάποια χαρακτηριστικά για το σηματοφόρο, αλλά συνήθως αφήνουμε τα default, δίνοντάς του την τιμή `NULL`
  - Ένας σηματοφόρος μπορεί να αρχικοποιηθεί και στατικά δίνοντάς του σαν τιμή τη σταθερά `PTHREAD_MUTEX_INITIALIZER`
  - Επιστρέφει πάντοτε 0

- Συνάρτηση βιβλιοθήκης `pthread_mutex_lock`
  - `int pthread_mutex_lock(pthread_mutex_t *mutex)`
  - Κλειδώνει το σηματοφόρο `*mutex` εφ΄ όσον είναι ξεκλείδωτος
  - Αν ο σηματοφόρος είναι κλειδωμένος από άλλο νήμα, τότε το καλούν νήμα τίθεται σε αναστολή και η συνάρτηση επιστρέφει όταν κατορθώσει να κλειδώσει το σηματοφόρο, αφού τον ξεκλειδώσει το νήμα που τον κλείδωσε
  - Επιστρέφει 0 σε επιτυχία
- Συνάρτηση βιβλιοθήκης `pthread_mutex_unlock`
  - `int pthread_mutex_unlock(pthread_mutex_t *mutex)`
  - Ξεκλειδώνει το σηματοφόρο `*mutex` εφ΄ όσον έχει κλειδωθεί προηγουμένως από το ίδιο νήμα
  - Επιστρέφει 0 σε επιτυχία
- Συνάρτηση βιβλιοθήκης `pthread_mutex_destroy`
  - `int pthread_mutex_destroy(pthread_mutex_t *mutex)`
  - Καταστρέφει το σηματοφόρο `*mutex` εφ΄ όσον είναι ξεκλείδωτος
  - Δεν είναι αναγκαίο να καταστρέφονται οι σηματοφόροι που έχουν αρχικοποιηθεί στατικά
  - Επιστρέφει 0 σε επιτυχία

- Χρήση των συναρτήσεων pthread_mutex_init, pthread_mutex_lock, pthread_mutex_unlock και pthread_mutex_destroy

```
/* File: sync_by_mutex.c */
#include <stdio.h>                                  /* For printf */
#include <string.h>                       /* For strcpy, strerror */
#include <stdlib.h>                                   /* For exit */
#include <pthread.h>                                /* For threads */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                             /* Error message printing function */
pthread_mutex_t mtx;                 /* Mutex for synchronization */
char buf[25];                           /* Message to communicate */
void *thread_f(void *);                    /* Forward declaration */

main()
{  pthread_t thr;
   int err;
   pthread_mutex_init(&mtx, NULL);
   if (err = pthread_mutex_lock(&mtx)) {          /* Lock mutex */
      perror2("pthread_mutex_lock", err); exit(1); }
   printf("Thread %d: Locked the mutex\n", pthread_self());
   if (err = pthread_create(&thr, NULL, thread_f, NULL)) {
                                                  /* New thread */
      perror2("pthread_create", err); exit(1); }
   printf("Thread %d: Created thread %d\n", pthread_self(), thr);
   strcpy(buf, "This is a test message");
   printf("Thread %d: Wrote message \"%s\" for thread %d\n",
         pthread_self(), buf, thr);
   if (err = pthread_mutex_unlock(&mtx)) {     /* Unlock mutex */
      perror2("pthread_mutex_unlock", err); exit(1); }
   printf("Thread %d: Unlocked the mutex\n", pthread_self());
   if (err = pthread_join(thr, NULL)) {      /* Wait for thread */
      perror2("pthread_join", err); exit(1); }  /* termination */
   printf("Thread %d: Thread %d exited\n", pthread_self(), thr);
   if (err = pthread_mutex_destroy(&mtx)) {   /* Destroy mutex */
      perror2("pthread_mutex_destroy", err); exit(1); }
   pthread_exit(NULL); }
```

```
void *thread_f(void *argp)                    /* Thread function */
{  int err;
   printf("Thread %d: Just started\n", pthread_self());
   printf("Thread %d: Trying to lock the mutex\n", pthread_self());
   if (err = pthread_mutex_lock(&mtx)) {         /* Lock mutex */
      perror2("pthread_mutex_lock", err); exit(1); }
   printf("Thread %d: Locked the mutex\n", pthread_self());
   printf("Thread %d: Read message \"%s\"\n", pthread_self(), buf);
   if (err = pthread_mutex_unlock(&mtx)) {     /* Unlock mutex */
      perror2("pthread_mutex_unlock", err); exit(1); }
   printf("Thread %d: Unlocked the mutex\n", pthread_self());
   pthread_exit(NULL); }
```

```
$ ./sync_by_mutex
Thread 1024: Locked the mutex
Thread 1026: Just started
Thread 1026: Trying to lock the mutex
Thread 1024: Created thread 1026
Thread 1024: Wrote message "This is a test message" for thread 1026
Thread 1024: Unlocked the mutex
Thread 1026: Locked the mutex
Thread 1026: Read message "This is a test message"
Thread 1026: Unlocked the mutex
Thread 1024: Thread 1026 exited
$
```

- *Να γραφεί ένα πρόγραμμα C που να δημιουργεί ένα δεδομένο αριθμό νημάτων, τα οποία να αναλαμβάνουν να υπολογίσουν, για δεδομένο n, το* $\sum_{i=1}^{n} i^2$*, έχοντας καθένα από τα νήματα την ευθύνη για τον υπολογισμό περίπου ίσου πλήθους όρων του αθροίσματος.*

```
/* File: sum_of_squares.c */
#include <stdio.h>                                      /* For printf */
#include <string.h>                                    /* For strerror */
#include <stdlib.h>                                       /* For exit */
#include <pthread.h>                                   /* For threads */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                                  /* Error message printing function */
pthread_mutex_t mtx;                     /* Mutex for synchronization */
int n, nthr, mtxfl;          /* Variables visible by thread function */
double sqsum;                                       /* Sum of squares */
void *square_f(void *);                        /* Forward declaration */

main(int argc, char *argv[])
{  int i, err;
   pthread_t *tids;
   if (argc > 3) {
      n = atoi(argv[1]);                /* Last integer to be squared */
      nthr = atoi(argv[2]);                      /* Number of threads */
      mtxfl = atoi(argv[3]); }               /* Are we going to lock? */
   else          exit(0);
   if (nthr > 50) {                          /* Avoid too many threads */
      printf("Number of threads should be up to 50\n"); exit(0); }
   if ((tids = malloc(nthr * sizeof(pthread_t))) == NULL) {
      perror("malloc"); exit(1); }
   sqsum = (double) 0.0;                             /* Initialize sum */
   pthread_mutex_init(&mtx, NULL);                /* Initialize mutex */
   for (i=0 ; i<nthr ; i++) {
      if (err = pthread_create(tids+i, NULL, square_f, (void *) i)) {
                                                   /* Create a thread */
         perror2("pthread_create", err); exit(1); } }
   for (i=0 ; i<nthr ; i++)
      if (err = pthread_join(*(tids+i), NULL)) {
                                      /* Wait for thread termination */
         perror2("pthread_join", err); exit(1); }
```

```
   if (err = pthread_mutex_destroy(&mtx)) {        /* Destroy mutex */
      perror2("pthread_mutex_destroy", err); exit(1); }
   if (!mtxfl)
      printf("Without mutex\n");
   else
      printf("With mutex\n");
   printf("%2d threads:    sum of squares up to %d is %12.9e\n",
         nthr, n, sqsum);
   sqsum = (double) 0.0;        /* Compute sum with a single thread */
   for (i=0 ; i<n ; i++)
      sqsum += (double) (i+1) * (double) (i+1);
   printf("Single thread: sum of squares up to %d is %12.9e\n", n, sqsum);
   printf("Formula based: sum of squares up to %d is %12.9e\n",
         n, ((double) n)*(((double) n)+1)*(2*((double) n)+1)/6);
   pthread_exit(NULL); }

void *square_f(void *argp)                          /* Thread function */
{  int i, thri, err;
   thri = (int) argp;
   for (i=thri ; i<n ; i+=nthr) {
      if (mtxfl)                                     /* Is mutex used? */
         if (err = pthread_mutex_lock(&mtx)) {        /* Lock mutex */
            perror2("pthread_mutex_lock", err); exit(1); }
         sqsum += (double) (i+1) * (double) (i+1);
      if (mtxfl)                                     /* Is mutex used? */
         if (err = pthread_mutex_unlock(&mtx)) {    /* Unlock mutex */
            perror2("pthread_mutex_unlock", err); exit(1); } }
   pthread_exit(NULL); }
```

```
$ ./sum_of_squares 123456 23 1
With mutex
23 threads:    sum of squares up to 123456 is 6.272210524e+14
Single thread: sum of squares up to 123456 is 6.272210524e+14
Formula based: sum of squares up to 123456 is 6.272210524e+14
$ ./sum_of_squares 123456 23 0
Without mutex
23 threads:    sum of squares up to 123456 is 5.740972751e+14
Single thread: sum of squares up to 123456 is 6.272210524e+14
Formula based: sum of squares up to 123456 is 6.272210524e+14
$
```

Σηματοφόροι και μεταβλητές συνθήκης

- Εκτός από τους δυαδικούς σηματοφόρους, η βιβλιοθήκη των POSIX νημάτων παρέχει και ένα δεύτερο μέσο συγχρονισμού νημάτων, τις μεταβλητές συνθήκης
- Ένα νήμα είναι δυνατόν, μέσω μίας μεταβλητής συνθήκης, να αναστείλει την εκτέλεσή του μέχρις ότου ικανοποιηθεί κάποια συνθήκη
- Η ικανοποίηση της συνθήκης γίνεται γνωστή στο νήμα που έχει αναστείλει την εκτέλεσή του από άλλο νήμα μέσω κατάλληλου μηχανισμού ενημέρωσης
- Μία μεταβλητή συνθήκης είναι πάντοτε συνυφασμένη με ένα δυαδικό σηματοφόρο
- Συνάρτηση βιβλιοθήκης `pthread_cond_init`
  - `int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *cond_attr)`
  - Αρχικοποιεί δυναμικά τη μεταβλητή συνθήκης `*cond`
  - Μέσω του `cond_attr` μπορούμε να ορίσουμε κάποια χαρακτηριστικά για τη μεταβλητή συνθήκης, αλλά συνήθως αφήνουμε τα default, δίνοντάς του την τιμή `NULL`
  - Μία μεταβλητή συνθήκης μπορεί να αρχικοποιηθεί και στατικά δίνοντάς της σαν τιμή τη σταθερά `PTHREAD_COND_INITIALIZER`
  - Επιστρέφει πάντοτε 0

- Συνάρτηση βιβλιοθήκης `pthread_cond_wait`
  - `int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)`
  - Ξεκλειδώνει το σηματοφόρο `*mutex` και αναστέλλει την εκτέλεση του καλούντος νήματος μέχρι να ενεργοποιηθεί και πάλι από κάποιο άλλο νήμα, μέσω της συνάρτησης `pthread_cond_signal` ή της `pthread_cond_broadcast`, των οποίων η περιγραφή θα ακολουθήσει, με βάση τη μεταβλητή συνθήκης `*cond`
  - Το καλούν νήμα πρέπει πριν την κλήση της `pthread_cond_wait` να έχει κλειδώσει το σηματοφόρο `*mutex`
  - Πριν επιστρέψει η `pthread_cond_wait`, κλειδώνει και πάλι το σηματοφόρο `*mutex`
  - Το νήμα που θα ενεργοποιήσει το καλούν νήμα πρέπει πριν την κλήση της συνάρτησης ενεργοποίησης, π.χ. της `pthread_cond_signal`, να κλειδώσει το σηματοφόρο `*mutex` και μετά την κλήση να τον ξεκλειδώσει
  - Επιστρέφει πάντοτε 0

- Συνάρτηση βιβλιοθήκης `pthread_cond_signal`
  - `int pthread_cond_signal(pthread_cond_t *cond)`
  - Ενεργοποιεί κάποιο νήμα που έχει αναστείλει την εκτέλεσή του με βάση τη μεταβλητή συνθήκης `*cond`
  - Επιστρέφει πάντοτε 0
- Συνάρτηση βιβλιοθήκης `pthread_cond_broadcast`
  - `int pthread_cond_broadcast(pthread_cond_t *cond)`
  - Ενεργοποιεί όλα τα νήματα που έχουν αναστείλει την εκτέλεσή τους με βάση τη μεταβλητή συνθήκης `*cond`
  - Επιστρέφει πάντοτε 0
- Συνάρτηση βιβλιοθήκης `pthread_cond_destroy`
  - `int pthread_cond_destroy(pthread_cond_t *cond)`
  - Καταστρέφει τη μεταβλητή συνθήκης `*cond`, εφ΄ όσον κανένα νήμα δεν έχει αναστείλει την εκτέλεσή του με βάση αυτήν
  - Δεν είναι αναγκαίο να καταστρέφονται οι μεταβλητές συνθήκης που έχουν αρχικοποιηθεί στατικά
  - Επιστρέφει 0 σε επιτυχία

- Χρήση των συναρτήσεων pthread_cond_init, pthread_cond_wait, pthread_cond_signal και pthread_cond_destroy

```
/* File: mutex_condvar.c */
#include <stdio.h>                                    /* For printf */
#include <string.h>                             /* For strcpy, strerror */
#include <stdlib.h>                                     /* For exit */
#include <pthread.h>                                  /* For threads */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                                  /* Error message printing function */
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
                                       /* Mutex for synchronization */
pthread_cond_t cvar;                            /* Condition variable */
char buf[25];                                /* Message to communicate */
void *thread_f(void *);                         /* Forward declaration */

main()
{  pthread_t thr; int err;
   pthread_cond_init(&cvar, NULL); /* Initialize condition variable */
   if (err = pthread_mutex_lock(&mtx)) {              /* Lock mutex */
      perror2("pthread_mutex_lock", err); exit(1); }
   printf("Thread %d: Locked the mutex\n", pthread_self());
   if (err = pthread_create(&thr, NULL, thread_f, NULL)) {
                                                      /* New thread */
      perror2("pthread_create", err); exit(1); }
   printf("Thread %d: Created thread %d\n", pthread_self(), thr);
   printf("Thread %d: Waiting for signal\n", pthread_self());
   pthread_cond_wait(&cvar, &mtx);                /* Wait for signal */
   printf("Thread %d: Woke up\n", pthread_self());
   printf("Thread %d: Read message \"%s\"\n", pthread_self(), buf);
   if (err = pthread_mutex_unlock(&mtx)) {          /* Unlock mutex */
      perror2("pthread_mutex_unlock", err); exit(1); }
   printf("Thread %d: Unlocked the mutex\n", pthread_self());
   if (err = pthread_join(thr, NULL)) {           /* Wait for thread */
      perror2("pthread_join", err); exit(1); }       /* termination */
   printf("Thread %d: Thread %d exited\n", pthread_self(), thr);
   if (err = pthread_cond_destroy(&cvar)) {
                                     /* Destroy condition variable */
      perror2("pthread_cond_destroy", err); exit(1); }
   pthread_exit(NULL); }
```

```
void *thread_f(void *argp)                          /* Thread function */
{  int err;
   printf("Thread %d: Just started\n", pthread_self());
   printf("Thread %d: Trying to lock the mutex\n", pthread_self());
   if (err = pthread_mutex_lock(&mtx)) {              /* Lock mutex */
      perror2("pthread_mutex_lock", err); exit(1); }
   printf("Thread %d: Locked the mutex\n", pthread_self());
   strcpy(buf, "This is a test message");
   printf("Thread %d: Wrote message \"%s\"\n", pthread_self(), buf);
   pthread_cond_signal(&cvar);                 /* Awake other thread */
   printf("Thread %d: Sent signal\n", pthread_self());
   if (err = pthread_mutex_unlock(&mtx)) {          /* Unlock mutex */
      perror2("pthread_mutex_unlock", err); exit(1); }
   printf("Thread %d: Unlocked the mutex\n", pthread_self());
   pthread_exit(NULL); }
```

```
$ ./mutex_condvar
Thread 1024: Locked the mutex
Thread 1026: Just started
Thread 1026: Trying to lock the mutex
Thread 1024: Created thread 1026
Thread 1024: Waiting for signal
Thread 1026: Locked the mutex
Thread 1026: Wrote message "This is a test message"
Thread 1026: Sent signal
Thread 1026: Unlocked the mutex
Thread 1024: Woke up
Thread 1024: Read message "This is a test message"
Thread 1024: Unlocked the mutex
Thread 1024: Thread 1026 exited
$
```

- *Να γραφεί ένα πρόγραμμα C που να δημιουργεί τέσσερα νήματα, τα τρία από τα οποία να αυξάνουν ένα καθολικό μετρητή, ενώ το τέταρτο να αναστέλλει την εκτέλεσή του μέχρι ο μετρητής να φτάσει συγκεκριμένη τιμή.*

```
/* File: counter.c */
#include <stdio.h>                                        /* For printf */
#include <stdlib.h>                                         /* For exit */
#include <pthread.h>                                     /* For threads */
#define perror2(s, e) fprintf(stderr, "%s: %s\n", s, strerror(e))
                                    /* Error message printing function */
#define COUNT_PER_THREAD 8          /* Count increments by each thread */
#define THRESHOLD  19                 /* Count value to wake up thread */
int count = 0;                                          /* The counter */
int thread_ids[4] = {0, 1, 2, 3};                     /* My thread ids */
pthread_mutex_t mtx;                                       /* My mutex */
pthread_cond_t cv;                              /* My condition variable */

void *incr(void *argp)
{  int i, j, err, *id = argp;
   for (i=0 ; i<COUNT_PER_THREAD ; i++) {
      if (err = pthread_mutex_lock(&mtx)) {              /* Lock mutex */
         perror2("pthread_mutex_lock", err); exit(1); }
      count++;                                       /* Increment counter */
      if (count == THRESHOLD) {                   /* Check for threshold */
         pthread_cond_signal(&cv);           /* Signal suspended thread */
         printf("incr: thread %d, count = %d, threshold reached\n",
               *id, count); }
      printf("incr: thread %d, count = %d\n", *id, count);
      if (err = pthread_mutex_unlock(&mtx)) {          /* Unlock mutex */
         perror2("pthread_mutex_unlock", err); exit(1); }
      for (j=0 ; j < 1000000 ; j++); }    /* For threads to alternate */
                                                      /* on mutex lock */
   pthread_exit(NULL); }
```

```
void *susp(void *argp)
{  int err, *id = argp;
   printf("susp: thread %d started\n", *id);
   if (err = pthread_mutex_lock(&mtx)) {                /* Lock mutex */
      perror2("pthread_mutex_lock", err); exit(1); }
   while (count < THRESHOLD) {            /* If threshold not reached */
      pthread_cond_wait(&cv, &mtx);                         /* suspend */
      printf("susp: thread %d, signal received\n", *id); }
   if (err = pthread_mutex_unlock(&mtx)) {            /* Unlock mutex */
      perror2("pthread_mutex_unlock", err); exit(1); }
   pthread_exit(NULL); }

main()
{  int i, err;
   pthread_t threads[4];
   pthread_mutex_init(&mtx, NULL);                /* Initialize mutex */
   pthread_cond_init(&cv, NULL);            /* and condition variable */
   for (i=0 ; i<3 ; i++)
      if (err = pthread_create(&threads[i], NULL, incr,
            (void *) &thread_ids[i])) {     /* Create threads 0, 1, 2 */
         perror2("pthread_create", err); exit(1); }
   if (err = pthread_create(&threads[3], NULL, susp,
         (void *) &thread_ids[3])) {               /* Create thread 3 */
      perror2("pthread_create", err); exit(1); }
   for (i=0 ; i<4 ; i++)
      if (err = pthread_join(threads[i], NULL)) {
         perror2("pthread_join", err); exit(1); };
                                     /* Wait for threads termination */
   printf("main: all threads terminated\n");
                              /* Destroy mutex and condition variable */
   if (err = pthread_mutex_destroy(&mtx)) {
      perror2("pthread_mutex_destroy", err); exit(1); }
   if (err = pthread_cond_destroy(&cv)) {
      perror2("pthread_cond_destroy", err); exit(1); }
   pthread_exit(NULL); }
```

```
$ ./counter
incr: thread 0, count = 1
incr: thread 1, count = 2
incr: thread 0, count = 3
incr: thread 2, count = 4
incr: thread 0, count = 5
incr: thread 1, count = 6
susp: thread 3 started
incr: thread 2, count = 7
incr: thread 1, count = 8
incr: thread 2, count = 9
incr: thread 1, count = 10
incr: thread 2, count = 11
incr: thread 1, count = 12
incr: thread 2, count = 13
incr: thread 0, count = 14
incr: thread 2, count = 15
incr: thread 0, count = 16
incr: thread 2, count = 17
incr: thread 0, count = 18
incr: thread 1, count = 19, threshold reached
incr: thread 1, count = 19
susp: thread 3, signal received
incr: thread 0, count = 20
incr: thread 1, count = 21
incr: thread 0, count = 22
incr: thread 2, count = 23
incr: thread 1, count = 24
main: all threads terminated
$
```

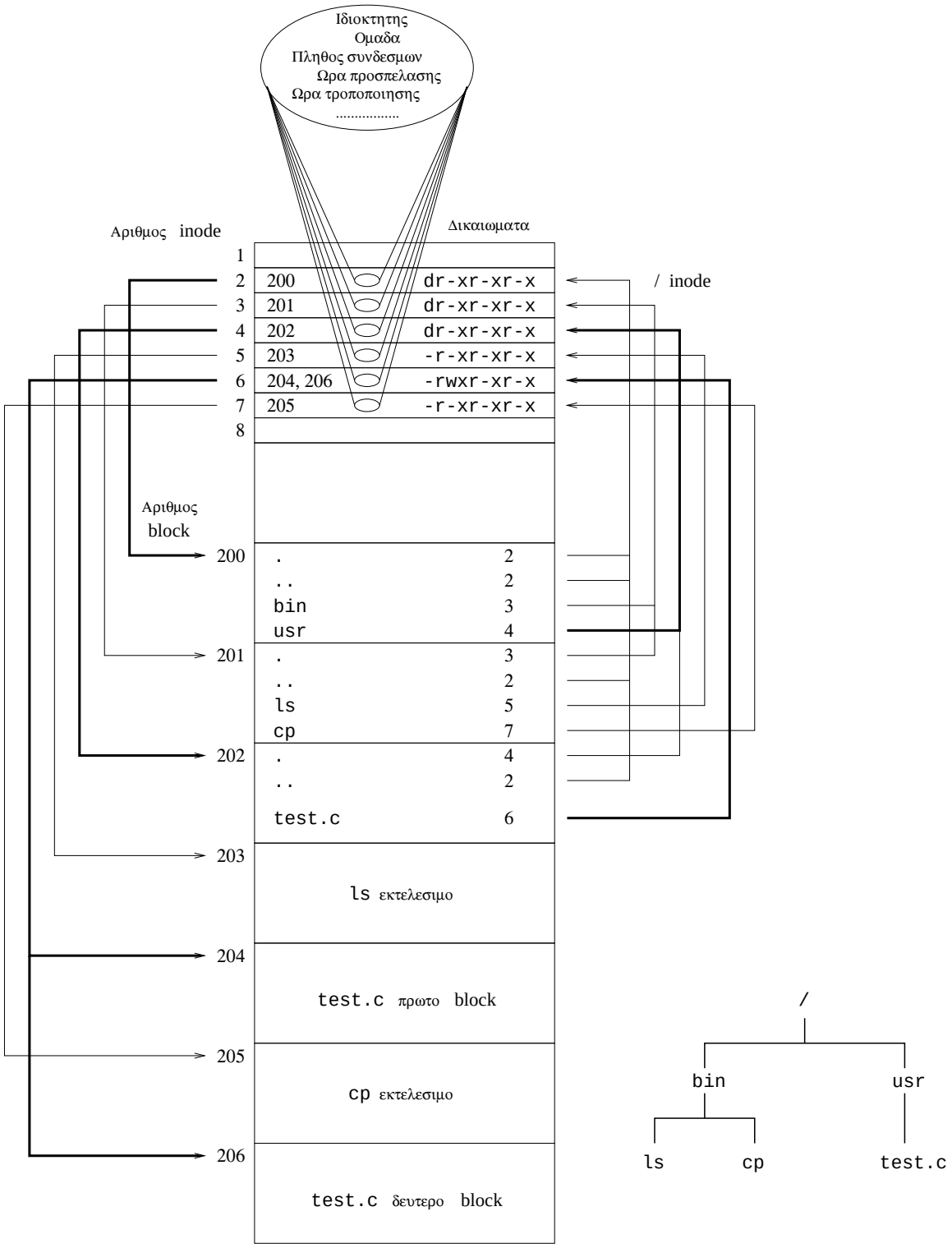
Άλλες ενδιαφέρουσες συναρτήσεις για τα νήματα

- pthread_equal
- pthread_attr_init
- pthread_attr_destroy
- pthread_attr_getdetachstate
- pthread_attr_setdetachstate
- pthread_mutex_trylock
- pthread_mutexattr_init
- pthread_mutexattr_destroy
- pthread_cond_timedwait
- pthread_condattr_init
- pthread_condattr_destroy
- pthread_key_create
- pthread_key_delete
- pthread_setspecific
- pthread_getspecific
- pthread_cancel
- pthread_setcancelstate
- pthread_setcanceltype
- pthread_testcancel
- ...

Διαχείριση συστήματος αρχείων

- Ιεραρχική οργάνωση του συστήματος αρχείων σε δευτερεύουσα μνήμη (δίσκος) μέσω ενός δέντρου που περιλαμβάνει
  - κοινά αρχεία
  - καταλόγους
  - συμβολικούς συνδέσμους
  - υποδοχές πεδίου Unix
  - συσκευές
  - ...
- Οι κατάλογοι είναι ενδιάμεσοι (όχι φύλλα) κόμβοι του δέντρου
- Για κάθε κόμβο του δέντρου, ένα σύνολο από απαραίτητες πληροφορίες (τύπος, δικαιώματα προστασίας, ιδιοκτήτης, ομάδα, ώρα τελευταίας τροποποίησης, διευθύνσεις blocks που βρίσκονται τα περιεχόμενα κ.λ.π.) είναι καταχωρημένες σε μία περιοχή του δίσκου που λέγεται inode του κόμβου
- Οι inodes όλων των κόμβων του συστήματος αρχείων είναι καταχωρημένοι στα πρώτα blocks του δίσκου
- Τα κοινά αρχεία περιέχουν μη δομημένα δεδομένα (ακολουθίες από bytes), ενώ οι κατάλογοι περιέχουν με συγκεκριμένη δομή ονόματα των κόμβων-παιδιών τους και αριθμούς των αντιστοίχων inodes

- *Πού βρίσκονται τα περιεχόμενα του αρχείου* `/usr/test.c`;



Πρόσβαση σε inodes

- Κλήση συστήματος `stat`
  - `int stat(char *path, struct stat *buf)`
  - Συμπληρώνει τα πεδία της δομής `*buf` με τις πληροφορίες που είναι καταχωρημένες στον inode του κόμβου με όνομα `path`
  - Κάποια πεδία στη δομή `struct stat *buf` είναι τα
    * `buf->st_ino`: Αριθμός inode του κόμβου
    * `buf->st_mode`: Τα 9 τελευταία bits κωδικοποιούν τα δικαιώματα προστασίας του ιδιοκτήτη, της ομάδας και των άλλων, κατά τα γνωστά, ενώ τα 4 πρώτα bits κωδικοποιούν τον τύπο του κόμβου (μετά από λογικό ΚΑΙ με τη σταθερά `S_IFMT`, αν το αποτέλεσμα είναι `S_IFDIR`, ο κόμβος είναι κατάλογος, αν είναι `S_IFREG`, είναι κοινό αρχείο, κ.λ.π.)
    * `buf->st_nlink`: Πλήθος σκληρών συνδέσμων προς τον κόμβο
    * `buf->st_uid`: Ταυτότητα ιδιοκτήτη
    * `buf->st_gid`: Ταυτότητα ομάδας
    * `buf->st_size`: Μέγεθος σε bytes
    * `buf->st_atime`: Ώρα τελευταίας προσπέλασης των περιεχομένων του κόμβου
    * `buf->st_mtime`: Ώρα τελευταίας τροποποίησης των περιεχομένων του κόμβου
    * `buf->st_ctime`: Ώρα τελευταίας τροποποίησης του inode του κόμβου

  - Στα πεδία `st_atime`, `st_mtime` και `st_ctime`, οι ώρες καταχωρούνται σαν πλήθος δευτερολέπτων που έχουν παρέλθει από την 1/1/1970
  - Η `stat` επιστρέφει 0 σε επιτυχία και -1 σε περίπτωση λάθους
  - Απαιτήσεις
    * `#include <sys/types.h>`
    * `#include <sys/stat.h>`

- Η συνάρτηση βιβλιοθήκης `ctime` μπορεί να χρησιμοποιηθεί για την μετατροπή των τιμών των πεδίων `st_atime`, `st_mtime` και `st_ctime` της δομής `struct stat` σε μία περισσότερο εύληπτη μορφή, παίρνοντας σαν όρισμα ένα δείκτη σε πλήθος δευτερολέπτων που έχουν παρέλθει από την 1/1/1970 και επιστρέφοντας έναν `char *` που δείχνει σε μία συμβολοσειρά η οποία περιγράφει την αντίστοιχη ημερομηνία και ώρα με τη μορφή που μας τη δίνει και η εντολή `date`

- Υπάρχει και αντίστοιχη με τη `stat` κλήση συστήματος `fstat`, η οποία αντί για `path` περιμένει έναν περιγραφέα αρχείου `int fd`

Πρόσβαση σε περιεχόμενα καταλόγων

- Τα περιεχόμενα καταλόγων (ζεύγη αριθμών inodes και ονομάτων κόμβων) είναι δυνατόν να προσπελασθούν μέσω των συναρτήσεων βιβλιοθήκης `opendir`, `readdir` και `closedir`
- Η πρόσβαση σε έναν κατάλογο γίνεται μέσω ενός δείκτη `DIR *` (ανάλογου με τον `FILE *` που χρησιμοποιείται στις συναρτήσεις διαχείρισης αρχείων της βιβλιοθήκης `stdio`)
- Κάθε στοιχείο στα περιεχόμενα ενός καταλόγου είναι μία δομή `struct dirent`, της οποίας τα πεδία `d_ino` και `d_name` δίνουν τον αριθμό inode και το όνομα, αντίστοιχα, ενός κόμβου που περιέχεται στον κατάλογο
- Δεν υπάρχει δυνατότητα, ούτε προγραμματιστικά, να τροποποιήσει ένας απλός χρήστης τα περιεχόμενα ενός καταλόγου επεμβαίνοντας στην εσωτερική δομή του
- Απαιτήσεις των συναρτήσεων βιβλιοθήκης για πρόσβαση στα περιεχόμενα καταλόγων
  - `#include <sys/types.h>`
  - `#include <sys/dirent.h>`

- Συνάρτηση βιβλιοθήκης `opendir`
  - `DIR *opendir(char *path)`
  - Ανοίγει τον κατάλογο με όνομα `path` και επιστρέφει έναν δείκτη σε `DIR` για την πρόσβαση στον κατάλογο
  - Επιστρέφει `NULL` σε περίπτωση λάθους
- Συνάρτηση βιβλιοθήκης `readdir`
  - `struct dirent *readdir(DIR *dp)`
  - Επιστρέφει έναν δείκτη σε δομή `struct dirent` που αντιστοιχεί στο τρέχον στοιχείο των περιεχομένων του καταλόγου που η πρόσβαση του γίνεται με το `dp`
  - Αν, για το τρέχον στοιχείο, το πεδίο `d_ino` της δομής είναι 0, ο αντίστοιχος κόμβος έχει διαγραφεί
  - Επιστρέφει `NULL` όταν δεν υπάρχουν άλλα στοιχεία για διάβασμα
- Συνάρτηση βιβλιοθήκης `closedir`
  - `int closedir(DIR *dp)`
  - Κλείνει τον κατάλογο που η πρόσβαση του γίνεται με το `dp`
  - Επιστρέφει 0 σε επιτυχία και -1 σε περίπτωση λάθους

- *Να γραφεί ένα πρόγραμμα C που να συμπεριφέρεται παρόμοια με την εντολή* `ls -al`.

```
/* File: myls.c */
#include <sys/types.h>          /* For stat, opendir, readdir, closedir */
#include <sys/stat.h>                                       /* For stat */
#include <dirent.h>                  /* For opendir, readdir, closedir */
#include <stdio.h>                                           /* For I/O */
#include <stdlib.h>                                  /* For malloc, exit */
#include <string.h>                        /* For strlen, strcpy, strcat */

char *modes[] = {"---", "--x", "-w-", "-wx", "r--", "r-x", "rw-", "rwx"};
                              /* Translate modes 0, 1, 2, 3, 4, 5, 6, 7 */
void list(char *);
void printout(char *);

main(int argc, char *argv[])
{  struct stat sbuf;
   if (argc < 2) {             /* If no arguments, list contents of "." */
      list("."); exit(0); }
   while (--argc) {                                  /* For each argument */
      if (stat(*++argv, &sbuf) < 0) {         /* Find inode information */
         perror(*argv); continue; }
      if ((sbuf.st_mode & S_IFMT) == S_IFDIR)
         list(*argv);                /* If directory, list its contents */
      else
         printout(*argv); } }   /* If regular file, print "ls -al" info */
```

```
void list(char *name)             /* Lists contents of directory "name" */
{  DIR *dp;
   struct dirent *dir;
   char *newname;
   if ((dp = opendir(name)) == NULL) {               /* Open directory */
      perror("opendir"); return; }
   while ((dir = readdir(dp)) != NULL) {      /* Read contents till end */
      if (dir->d_ino == 0)                    /* Ignore removed entries */
         continue;
      newname = malloc(strlen(name)+strlen(dir->d_name)+2);
      strcpy(newname, name);  /* Compose pathname as "name/dir->d_name" */
      strcat(newname, "/");
      strcat(newname, dir->d_name);
      printout(newname);                /* Print "ls -al" info of entry */
      free(newname); }
   closedir(dp); }                                  /* Close directory */

void printout(char *name)            /* Prints "ls -al" info of "name" */
{  struct stat sbuf;
   char type, perms[10];
   int i, j;
   stat(name, &sbuf);                           /* Find inode information */
   switch (sbuf.st_mode & S_IFMT) {
      case S_IFREG: type = '-'; break;                 /* Regular file */
      case S_IFDIR: type = 'd'; break;                    /* Directory */
      default:      type = '?'; break; }            /* Everything else */
   *perms = '\0';
   for (i=2 ; i >= 0 ; i--) {     /* Owner, group and other permissions */
      j = (sbuf.st_mode >> (i*3)) & 07;
      strcat(perms, modes[j]); }
   printf("%c%s%3d %5d/%-5d %7d %.12s %s\n", type, perms,
          sbuf.st_nlink, sbuf.st_uid, sbuf.st_gid, sbuf.st_size,
          ctime(&sbuf.st_mtime)+4, name); }         /* Print everything */
```

```
$ ./myls .. ../sh_scripts i*.c /usr/share/man
drwx------  6 32479/100      4096 Feb 21 11:43 ../.
drwxr-xr-x  8     0/0        4096 Feb  2 09:15 ../..
drwx------  2 32479/100      4096 Feb 19 11:46 ../.ssh
-rw-r--r--  1 32479/100       820 Jan 27 12:10 ../.emacs
-rw-r--r--  1 32479/100        24 Jan 27 12:10 ../.bash_logout
-rw-r--r--  1 32479/100       191 Jan 27 12:10 ../.bash_profile
-rw-r--r--  1 32479/100       134 Jan 27 12:24 ../.bashrc
-rw-r--r--  1 32479/100      3511 Jan 27 12:10 ../.screenrc
-rw-------  1 32479/100     15845 Feb 20 15:38 ../.bash_history
drwxr-xr-x  2 32479/100      4096 Jan 27 12:56 ../bin
drwxr-xr-x  2 32479/100      4096 Feb 12 15:12 ../sh_scripts
drwxr-xr-x  2 32479/100      4096 Feb 20 15:26 ../c_progs
drwxr-xr-x  2 32479/100      4096 Feb 12 15:12 ../sh_scripts/.
drwx------  6 32479/100      4096 Feb 21 11:43 ../sh_scripts/..
-rwxr-xr-x  1 32479/100       162 Feb 12 15:05 ../sh_scripts/factorial
-rwxr-xr-x  1 32479/100       202 Feb 10 22:37 ../sh_scripts/lsdir
-rwxr-xr-x  1 32479/100       500 Feb  7 21:04 ../sh_scripts/math
-rwxr-xr-x  1 32479/100       459 Feb 12 15:12 ../sh_scripts/maxsize
-rwxr-xr-x  1 32479/100       238 Feb 12 15:02 ../sh_scripts/once
-rwxr-xr-x  1 32479/100       304 Feb 12 15:08 ../sh_scripts/revstrs
-rw-r--r--  1 32479/100      2431 Feb 19 14:34 int_dgr_client.c
-rw-r--r--  1 32479/100      2086 Feb 19 14:25 int_dgr_server.c
-rw-r--r--  1 32479/100      2325 Feb 19 14:35 int_str_client.c
-rw-r--r--  1 32479/100      3138 Feb 19 14:36 int_str_server.c
-rw-r--r--  1 32479/100       862 Feb 19 12:08 io_demo.c
drwxr-xr-x 15     0/0        4096 Dec 27 14:37 /usr/share/man/.
drwxr-xr-x 74     0/0        4096 Dec 27 14:55 /usr/share/man/..
drwxr-xr-x  2     0/0        4096 Feb 16 15:16 /usr/share/man/man4
drwxr-xr-x  2     0/0       24576 Feb 16 15:16 /usr/share/man/man1
drwxr-xr-x  2     0/0        8192 Feb 16 15:16 /usr/share/man/man2
drwxr-xr-x  2     0/0       53248 Feb 16 15:16 /usr/share/man/man3
drwxr-xr-x  2     0/0        8192 Feb 16 15:16 /usr/share/man/man5
drwxr-xr-x  2     0/0        4096 Feb 16 15:16 /usr/share/man/man6
drwxr-xr-x  2     0/0        4096 Feb 16 15:16 /usr/share/man/man7
drwxr-xr-x  2     0/0        8192 Feb 16 15:16 /usr/share/man/man8
drwxr-xr-x  2     0/0        4096 May 15 22:38 /usr/share/man/man9
drwxr-xr-x  2     0/0        8192 Dec 27 14:30 /usr/share/man/mann
drwxr-xr-x  3     0/0        4096 Dec 27 14:21 /usr/share/man/pt_BR
drwxr-xr-x  4     0/0        4096 Dec 27 14:30 /usr/share/man/ja
drwxr-xr-x  3     0/0        4096 Dec 27 14:34 /usr/share/man/man
-rw-r--r--  1     0/0        1315 Jun 25 16:02 /usr/share/man/tmac.h
$
```

Άλλες λειτουργίες στο σύστημα αρχείων

- Κλήσεις συστήματος `unlink` και `link`
  - `int unlink(char *path)`
  - `int link(char *oldpath, char *newpath)`
  - Η `unlink` αποσυνδέει τα περιεχόμενα του κόμβου με όνομα `path` (δηλαδή τον αριθμό inode που αντιστοιχεί σ΄ αυτόν) από το όνομα αυτό
  - Η `link` συνδέει τον αριθμό inode που αντιστοιχεί στο όνομα `oldpath` και με το όνομα `newpath`
- Κλήσεις συστήματος `mkdir` και `rmdir`
  - `int mkdir(char *path, int mode)`
  - `int rmdir(char *path)`
  - Η `mkdir` δημιουργεί ένα νέο κατάλογο με όνομα `path` και δικαιώματα προστασίας `mode` (δικαιώματα που δεν επιτρέπονται από την τρέχουσα τιμή του `umask` δεν δίνονται στον κατάλογο)
  - Η `rmdir` διαγράφει τον κατάλογο με όνομα `path`, εφ΄ όσον είναι κενός

- Κλήση συστήματος `rename`
  - `int rename(char *oldpath, char *newpath)`
  - Μετονομάζει τον κόμβο με όνομα `oldpath` ώστε να έχει όνομα `newpath`
- Κλήση συστήματος `chmod`
  - `int chmod(char *path, int mode)`
  - Αλλάζει τα δικαιώματα προστασίας του κόμβου με όνομα `path` σε αυτά που περιγράφονται από το `mode` (κατά τα γνωστά)
  - Υπάρχει και αντίστοιχη κλήση συστήματος `fchmod`, η οποία αντί για `path` περιμένει έναν περιγραφέα αρχείου `int fd`
- Κλήσεις συστήματος `symlink` και `readlink`
  - `int symlink(char *oldpath, char *newpath)`
  - `int readlink(char *path, char *buf, int size)`
  - Η `symlink` δημιουργεί έναν νέο κόμβο με όνομα `newpath` που είναι συμβολικός σύνδεσμος και δείχνει στο όνομα `oldpath`
  - Η `readlink` επιστρέφει στο `buf`, που είναι μεγέθους `size` bytes, το όνομα στο οποίο δείχνει ο συμβολικός σύνδεσμος με όνομα `path`, ενώ στο όνομά της επιστρέφει το πλήθος των bytes που καταχωρήθηκαν στο `buf`
- Όλες οι προηγούμενες κλήσεις συστήματος για τις λειτουργίες στο σύστημα αρχείων επιστρέφουν -1 σε περίπτωση λάθους

Βιβλιογραφία

- G. Glass: "*Unix for Programmers and Users*", Prentice-Hall, 1993.
- M. J. Rochkind: "*Advanced Unix Programming*", 2nd Edition, Addison-Wesley, 2004.
- W. R. Stevens, S. A. Rago: "*Advanced Programming in the Unix Environment*", 2nd Edition, Addison-Wesley, 2005.
- D. A. Curry: "*Using C on the Unix System*", O'Reilly & Associates, 1991.
- K. Haviland, D. Gray, B. Salama: "*Unix System Programming*", Addison-Wesley, 1999.
- W. R. Stevens: "*Unix Network Programming*", Prentice-Hall, 1990.
- D. R. Butenhof: "*Programming with POSIX Threads*", Addison-Wesley, 1997.
- Lawrence Livermore National Laboratory: "*POSIX Threads Programming*", `http://www.llnl.gov/computing/tutorials/pthreads/`, 2003.

Ευχαριστίες ...

- ... στο *Στέφανο Σταμάτη*, μεταπτυχιακό φοιτητή του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Αθηνών, για την πολύπλευρη υποστήριξή του στο μάθημα από την άνοιξη του 2000 και μετά (σχεδίαση και υλοποίηση του επί πολλά χρόνια δυναμικού ιστοχώρου του μαθήματος, διαρκής επίλυση αποριών και προβλημάτων στους συναδέλφους του, υπόδειξη λαθών στις σημειώσεις και τα προγράμματα του μαθήματος και συνεχείς προτάσεις για τη βελτίωσή τους, κ.λ.π.).
- ... στο *Νίκο Καραμπατζιάκη*, απόφοιτο του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Αθηνών, και τώρα μεταπτυχιακό φοιτητή του Cornell University, για την περιεκτική περίληψη που συνέταξε το φθινόπωρο του 2002 σχετικά με τις προγραμματιστικές δυνατότητες που προσφέρονται από τα νήματα, η οποία ήταν μία από τις πηγές επάνω στις οποίες βασίστηκε το σχετικό τμήμα των σημειώσεων αυτών, καθώς επίσης και για το χρόνο που διέθεσε για να συζητήσουμε τα σχετικά θέματα.
- ... last, but not least, σε όλες τις φοιτήτριες και όλους τους φοιτητές που παρακολούθησαν το μάθημα αυτό που δίδαξα για 14 χρόνια (σε 15 συνολικά εξάμηνα), για τα πάντοτε εποικοδομητικά σχόλια και παρατηρήσεις που έκαναν, που πιστεύω ότι βελτίωσαν σε πολύ μεγάλο βαθμό την ποιότητα του περιεχομένου του, για την αναγνώριση της χρησιμότητας αυτού του μαθήματος στις σπουδές τους, όπως την εξέφραζαν επίσημα και ανεπίσημα με κάθε αφορμή, αλλά και για την ανυπόκριτη εκτίμηση και τον ειλικρινή σεβασμό που έδειχναν πάντοτε στο πρόσωπό μου. Πραγματικά, σας ευχαριστώ θερμότατα!

Παναγιώτης Σταματόπουλος