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>`

Παράδειγμα χρήσης perror, errno (πρόγραμμα errors_demo.c)

```
#include <stdio.h> /* For fopen, printf */
#include <stdlib.h> /* For malloc */
#include <errno.h> /* For errno variable */

int 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(40000000000U);
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");
}
return 1;
}
```

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

Είσοδος/Εξοδος 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

Είσοδος/Εξοδος Χαμηλού Επιπέδου (συν.)

- 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

  ↑ Αυτό αργότερα στο εξάμηνο

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:`	Άνοιγμα για διάβασμα μόνο
`O_WRONLY:`	Άνοιγμα για γράψιμο μόνο
`O_RDWR:`	Άνοιγμα 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

Παραδείγματα χρήσης open

```
int myfd;
myfd = open("/home/ann/my.dat", O_RDONLY);

mode_t fdmode = (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);

if ((fd = open("info.dat", O_RDWR | O_CREAT, fdmode)) == -1)
   perror("Failed to open info.dat");
```

S_IRUSR	Ανάγνωση ιδιοκτήτη
S_IWUSR	Γραφή ιδιοκτήτη
S_IXUSR	Εκτέλεση ιδιοκτήτη
S_IRWXU	Ανάγνωση, γραφή, εκτέλεση ιδιοκτήτη
S_IRGRP	Ανάγνωση ομάδα
S_IWGRP	Γραφή ομάδα
S_IXGRP	Εκτέλεση ομάδα
S_IRWXG	Ανάγνωση, γραφή, εκτέλεση ομάδα
S_IROTH	Ανάγνωση λοιποί
S_IWOTH	Γραφή λοιποί
S_IXOTH	Εκτέλεση λοιποί
S_IRWXO	Ανάγνωση, γραφή, εκτέλεση λοιποί

Εντολή read

- `int read(int fd, char *buf, int count)`
- Διαβάζει το πολύ count bytes από την οντότητα (αρχείο, συσκευή, άφρο 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

Πότε διαβάζει λιγότερα από count bytes?

- Αν το αρχείο έχει λιγότερα bytes
- Αν διακοπεί από σήμα
- Αν διαβάζει από σωλήνωση/υποδοχή, διαβάζει όταν βρίσκει κάτι διαθέσιμο => απαιτεί while loop για διάβασμα όλων των χαρακτήρων

Παράδειγμα συνάρτησης read

```
char buf[100];
ssize_t bytesread;

bytesread = read(STDIN_FILENO, buf, 100);
```

```
char *buf;
ssize_t bytesread;

bytesread = read(STDIN_FILENO, buf, 100);
```

Τι θα συμβεί εδώ? Στα επόμενα παραδείγματα, STDIN_FILENO=0,

STDOUT_FILE=1,

STDERR_FILE=2

Εντολές write, close

- `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

- `int close(int fd)`
- Ελευθερώνει τον περιγραφέα `fd`
- Επιστρέφει 0 σε επιτυχία ή -1 σε αποτυχία

Κάντε man στις εντολές read, write, close για τα include files. Συχνά το unistd.h

Παραδείγματα εντολής write

```
#define BLKSIZE 1024
char buf[BLKSIZE];

read(STDIN_FILENO, buf, BLKSIZE);
write(STDOUT_FILENO, buf, BLKSIZE);
```

Ίσως διαβάσει < 1024 bytes

```
#define BLKSIZE 1024
char buf[BLKSIZE];
ssize_t bytesread;

bytesread = read(STDIN_FILENO, buf, BLKSIZE);
if (bytesread > 0)
   write(STDOUT_FILE, buf, bytesread);
```

Καμιά εγγύηση ότι θα γράψει bytesread bytes

Παράδειγμα χρήσης read, write, close

```
#include <stdio.h> /* For printf */
#include <stdlib.h> /* For exit */
#include <fcntl.h>
#include <unistd.h>

int 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); return 1;}
```

```
$ ./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 dup, 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

Παράδειγμα dup, 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 */
#include <unistd.h>

int 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); return 1;}
```

```
$ ./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
```

Παράδειγμα

Γράψτε ένα πρόγραμμα που αντιγράφει ένα αρχείο στο τέλος ενός άλλου

```
#include <string.h> /* For strlen */
#include <stdlib.h> /* For exit */
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

int 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[0], strlen(argv[0]));
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);
return 1; } /* Close to-file */
```

Πρόσβαση σε inodes – Κλήση 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

Πρόσβαση σε inodes – Κλήση stat (συν.)

- 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`

Παράδειγμα stat

```
#include <stdio.h>
#include <time.h>
#include <sys/stat.h>

void printaccessmodbad(char *path) {
   struct stat statbuf;

   if (stat(path, &statbuf) == -1)
      perror("Failed to get file status");
   else
     printf("%s accessed: %s modified: %s", path,
           ctime(&statbuf.st_atime), ctime(&statbuf.st_mtime));
}
```

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>`

Συναρτήσεις opendir, readdir, closedir

- 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

Παράδειγμα opendir, readdir, closedir

```
#include <dirent.h>
#include <errno.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
   struct dirent *direntp;
   DIR *dirp;

   if (argc != 2) {
      fprintf(stderr, "Usage: %s directory_name\n", argv[0]);
      return 1;
   }

   if ((dirp = opendir(argv[1])) == NULL) {
      perror ("Failed to open directory");
      return 1;
   }

   while ((direntp = readdir(dirp)) != NULL)
      printf("%s\n", direntp->d_name);
   while ((closedir(dirp) == -1) && (errno == EINTR)) ;
   return 0;
}
```

Παράδειγμα- Πρόγραμμα με συμπεριφορά 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 */
```

Συνέχεια παραδείγματος

```
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 */
```

Κλήσεις link, unlink, mkdir, rmdir

- 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 ← Όχι διαγραφή αρχείου αν counter > 0
  - 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

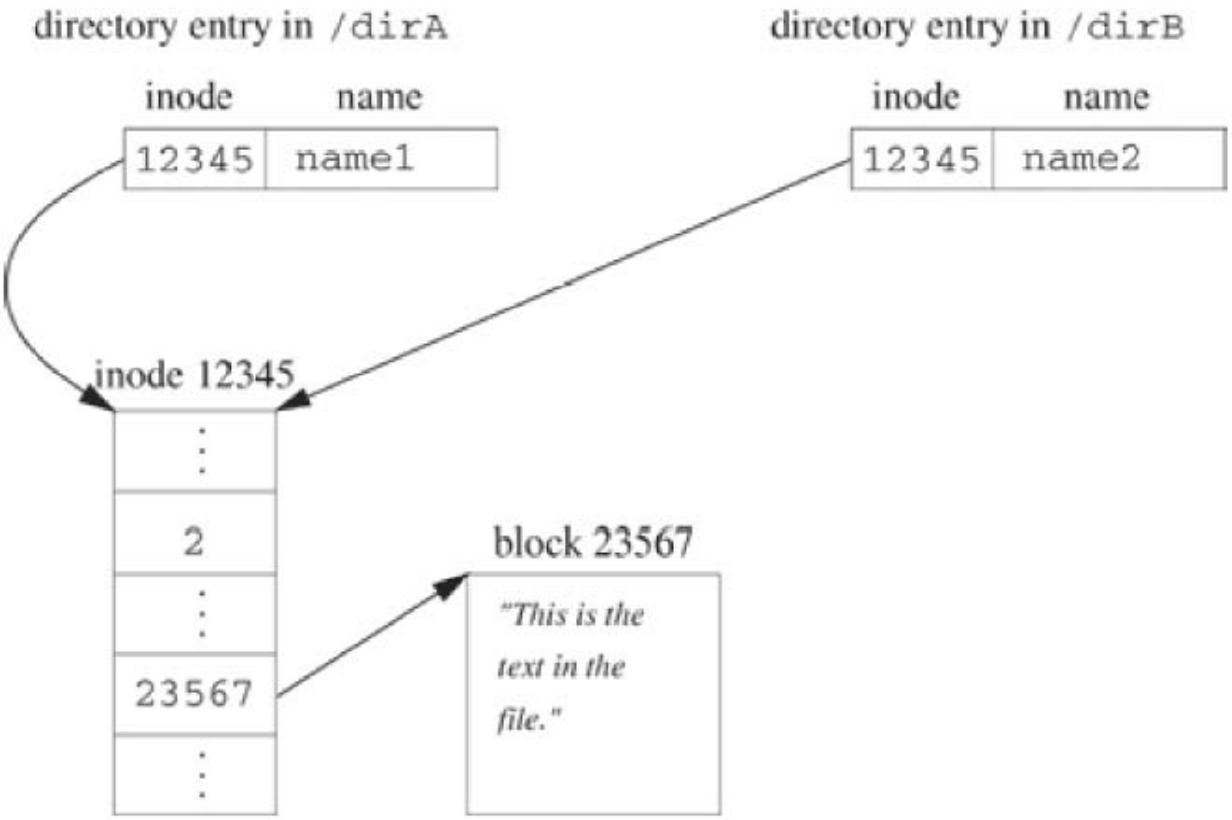
Κλήσεις rename, chmod, symlink, readlink,

- 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

Παράδειγμα link

```
#include <stdio.h>
#include <unistd.h>

if (link("/dirA/name1", "/dirB/name2") == -1)
   perror("Failed to make a new link in /dirB");
```



Σήματα

- Ειδοποίηση μέσω λογισμικού σε διαδικασία ότι κάτι συνέβη
  - Σφάλμα εκτέλεσης, core
  - Από πληκτρολόγιο (Ctrl-C, Ctrl-Z)
  - Μια διεργασία σε μία άλλη
  - Εντολή kill κτλ…
- 4 Επιλογές για κάθε σήμα:
  - Μπλοκάρισέ το
  - Αγνόησέ το
  - Προκαθορισμένη δράση
  - Ορισμός συνάρτησης (διαχειριστής σήματος – signal handler) για το πώς να αντιμετωπιστεί

Πόσα Σήματα Υπάρχουν? (1)

signal	description	default action
SIGABRT	process abort	implementation dependent
SIGALRM	alarm clock ← Μετρητής	abnormal termination
SIGBUS	access undefined part of memory object	implementation dependent
SIGCHLD	child terminated, stopped or continued	ignore
SIGCONT	execution continued if stopped	continue
SIGFPE	error in arithmetic operation as in division by zero	implementation dependent
SIGHUP	hang-up (death) on controlling terminal (process)	abnormal termination
SIGILL	invalid hardware instruction	implementation dependent
SIGINT	interactive attention signal (usually Ctrl-C) ← Σημαντικό	abnormal termination
SIGKILL	terminated (cannot be caught or ignored) ← Σημαντικό	abnormal termination
SIGPIPE	write on a pipe with no readers	abnormal termination

Πόσα Σήματα Υπάρχουν? (2)

signal	description	default action
SIGQUIT	interactive termination: core dump (usually Ctrl-\)	implementation dependent
SIGSEGV	invalid memory reference	implementation dependent
SIGSTOP	execution stopped (cannot be caught or ignored)	stop
SIGTERM	termination	abnormal termination
SIGTSTP	terminal stop	stop
SIGTTIN	background process attempting to read	stop
SIGTTOU	background process attempting to write	stop
SIGURG	high bandwidth data available at a socket	ignore
SIGUSR1	user-defined signal 1	abnormal termination
SIGUSR2	user-defined signal 2	abnormal termination

Ctrl-\

Σημαντικό

Συχνά Ctrl-Z

Σημαντικό

Αν χρησιμοποιείτε το όνομα κάποιου σήματος, include <signal.h>

Εντολή kill

Εντολή κελύφους:

kill [-signalName] processId

kill -s signalName processId

kill -l (το γράμμα l εδώ, όχι το νούμερο)

Παραδείγματα:

kill -9 3424

kill -s USR1 3424

kill -9 3424

- 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

Εντολές raise, pause

- Εντολή int pause()
  - Αναμονή διεργασίας μέχρι τη λήψη σήματος που η δράση της είτε εκτελεί συνάρτηση χρήστη είτε τερματίζει διεργασία
  - Πάντα επιστρέφει -1
  - #include <unistd.h>
- Εντολή int raise(int signalId)
  - Στέλνει σήμα στην ίδια διεργασία!!!
  - Σε επιτυχία 0. Αλλιώς <> 0 και θέτει το errno

Πχ:

```
if (raise(SIGUSR1) != 0)
    perror("Failed to raise SIGUSR1");
```

Συνάρτηση alarm

- 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`

#include <unistd.h>

```
/* 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
$
```

Συνάρτηση signal

Παλιός τρόπος ορισμού συνάρτησης χειρισμού σημάτων

ΜΗΝ την χρησιμοποιείτε

- `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
- Η `signal` επιστρέφει τον προηγούμενο διαχειριστή σήματος σε επιτυχία ή -1 σε αποτυχία

Μάσκες σημάτων

- Ορισμός δράσεων για πολλά σήματα μαζί με χρήση μάσκας σήματος.
  - Δυαδικός αριθμός με κατάλληλα bits σε 1
  - sigemptyset: αρχικοποίηση – κανένα σήμα ορισμένο
  - sigfillset: αρχικοποίηση – όλα τα σήματα ορισμένα
  - sigaddset/sigdelset: εισαγωγή/διαγραφή σήματος από μάσκα
  - sigmember: υπάρχει το σήμα στη μάσκα?

```
#include <signal.h>

int sigaddset(sigset_t *set, int signo);
int sigdelset(sigset_t *set, int signo);
int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);
int sigismember(const sigset_t *set, int signo);
```

Παραδείγματα Χειρισμού Μάσκας Σημάτων (1)

- Πώς να βάλουμε σε μία μάσκα τα σήματα SIGQUIT και SIGINT

```
if ((sigemptyset(&twosigs) == -1) ||
    (sigaddset(&twosigs, SIGINT) == -1)  ||
    (sigaddset(&twosigs, SIGQUIT) == -1))
    perror("Failed to set up signal mask");
```

Παραδείγματα Χειρισμού Μάσκας Σημάτων (2)

Πώς να ορίσουμε σήματα για αναμονή

Τα σήματα θα παραδοθούν όταν καταργήσουμε την αναμονή τους

int sigprocmask(int how, const sigset_t *restrict set, sigset_t *restrict oset)

If oset<>NULL, μετά την κλήση το oset περιέχει την προηγούμενη μάσκα

how =

SIG_BLOCK: Προσθήκη σημάτων σε ήδη μπλοκαρισμένα

SIG_UNBLOCK: Αφαίρεση σημάτων από ήδη μπλοκαρισμένα

SIG_SETMASK: Ορίζει τα μπλοκαρισμένα σήματα

```
sigset_t newsigset;

if ((sigemptyset(&newsigset) == -1) ||
    (sigaddset(&newsigset, SIGINT) == -1))
   perror("Failed to initialize the signal set");
else if (sigprocmask(SIG_BLOCK, &newsigset, NULL) == -1)
   perror("Failed to block SIGINT");
```

Σε αναμονή το SIGINT

Παραδείγματα Χειρισμού Μάσκας Σημάτων (3)

Βάζοντας το SIGINT προσωρινά σε αναμονή

```
#include <math.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc,  char *argv[]) {
    int i;
    sigset_t intmask;
    int repeatfactor;
    double y = 0.0;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s repeatfactor\n", argv[0]);
        return 1;
    }
    repeatfactor = atoi(argv[1]);
    if ((sigemptyset(&intmask) == -1) || (sigaddset(&intmask, SIGINT) == -1)){
      perror("Failed to initialize the signal mask");
      return 1;
  }
  for ( ; ; ) {
      if (sigprocmask(SIG_BLOCK, &intmask, NULL) == -1)
          break;
      fprintf(stderr, "SIGINT signal blocked\n");
      for (i = 0; i < repeatfactor; i++)
          y += sin((double)i);
      fprintf(stderr, "Blocked calculation is finished, y = %f\n", y);
      if (sigprocmask(SIG_UNBLOCK, &intmask, NULL) == -1)
          break;
      fprintf(stderr, "SIGINT signal unblocked\n");
      for (i = 0; i < repeatfactor; i++)
          y += sin((double)i);
      fprintf(stderr, "Unblocked calculation is finished, y=%f\n", y);
  }
  perror("Failed to change signal mask");
  return 1;
}
```

Σε αναμονή το SIGINT

Τέλος αναμονής. Αν είχε πατηθεί πριν Ctrl-C, τερματισμός προγράμματος

Πιάνοντας και αγνοώντας σήματα - sigaction

Επιτρέπει να δούμε ή να ορίσουμε τη δράση σχετικά με κάποιο σήμα.

```
#include <signal.h>

int sigaction(int sig, const struct sigaction *restrict act,
              struct sigaction *restrict oact);
```

Επιστρέφει 0 σε επιτυχία. Σε αποτυχία -1 και θέτει το errno (EINVAL συνήθως). If oact<>NULL, μετά την κλήση το oact περιέχει την προηγούμενη δράση για το σήμα

Προκαθορισμένη δράση

Αγνόησέ το

```
struct sigaction {
  void (*sa_handler)(int); /* SIG_DFL, SIG_IGN or pointer to function */
  sigset_t sa_mask;        /* additional signals to be blocked
                              during execution of handler */
  int sa_flags;            /* special flags and options */
  void(*sa_sigaction) (int, siginfo_t *, void *); /* realtime handler */
};
```

ΠΡΟΣΟΧΗ: Να ορίζετε μόνο ένα από τα sa_handler και sa_sigaction.

Παράδειγμα sigaction

Ορίζουμε τη συνάρτηση mysighand να καλείται στη λήψη SIGINT

```
struct sigaction newact;


newact.sa_handler = mysighand;  /* set the new handler */
newact.sa_flags = 0;            /* no special options */
if ((sigemptyset(&newact.sa_mask) == -1) ||
    (sigaction(SIGINT, &newact, NULL) == -1))
    perror("Failed to install SIGINT signal handler");
```

Παράδειγμα sigaction (2)

Πώς να αγνοήσουμε το SIGINT αν η ισχύει η προκαθορισμένη δράση για αυτό

```
struct sigaction act;

if (sigaction(SIGINT, NULL, &act) == -1)  /* Find current SIGINT handler */
   perror("Failed to get old handler for SIGINT");
else if (act.sa_handler == SIG_DFL) {    /* if SIGINT handler is default */
   act.sa_handler = SIG_IGN;         /* set new SIGINT handler to ignore */
   if (sigaction(SIGINT, &act, NULL) == -1)
      perror("Failed to ignore SIGINT");
}
```

NULL σημαίνει κάνω ερώτηση για τρέχουσα δράση

Παράδειγμα sigaction (3)

Ας ορίσουμε και μια συνάρτηση αντιμετώπισης του SIGINT…

```
void catchctrlc(int signo) {
   char handmsg[] = "I found Ctrl-C\n";
   int msglen = sizeof(handmsg);

   write(STDERR_FILENO, handmsg, msglen);
}
...
struct sigaction act;
act.sa_handler = catchctrlc;
act.sa_flags = 0;
if ((sigemptyset(&act.sa_mask) == -1) ||
    (sigaction(SIGINT, &act, NULL) == -1))
   perror("Failed to set SIGINT to handle Ctrl-C");
```

Πάντα write σε χειριστές σημάτων

Παράδειγμα mytest.c

```
#include        <stdio.h>
#include        <string.h>
#include        <curses.h>
#include        <signal.h>
#include        <unistd.h>
#include        <errno.h>
char buf[128];

void move_msg(int signum)
{
        signal(SIGALRM, SIG_IGN);       /* reset, just in case  */
        signal(SIGINT, move_msg);       /* reset, just in case  */
        strcpy(buf, "Received Alarm\n");
        write(1, buf, strlen(buf));
}

int main() {
        int delay, ndelay, c;
        char ch;

        signal(SIGALRM, SIG_IGN);       /* reset, just in case  */
        signal(SIGINT, move_msg);       /* reset, just in case  */
        set_ticker( 500 );

        while(1) {
                ndelay = 0;
                c = read(0, &ch, sizeof(char));
                sprintf(buf, "Read char = %c and errno = %d and EINTR = %d\n",
                        ch, errno, EINTR);
                write(1, buf, strlen(buf));
                if ( c == 'Q' ) break;
                if ( ndelay > 0 )
                        set_ticker( delay = ndelay );
        }
        return 0;
}
```

Επαναλαμβανόμενο ρολόι. Συνάρτηση στο ticker_demo.c

Μετά από κάθε Ctrl-C επιστρέφει εδώ. Αγνοεί το SIGALARM

Παράδειγμα mytest2.c

```
#include        <stdio.h>
#include        <string.h>
#include        <curses.h>
#include        <signal.h>
#include        <unistd.h>
#include        <errno.h>
char buf[128];

struct sigaction act1, act2;

void move_msg(int signum)
{
        act1.sa_handler = SIG_IGN;
        act2.sa_handler = move_msg;
        sigaction(SIGALRM, &act1, NULL);        /* reset, just in case  */
        sigaction(SIGINT, &act2, NULL);         /* reset, just in case  */
        strcpy(buf, "Received Alarm\n");
        write(1, buf, strlen(buf));
}

int main() {
        int delay, ndelay, c;
        char ch;

        act1.sa_handler = SIG_IGN;
        act2.sa_handler = move_msg;
        sigaction(SIGALRM, &act1, NULL);        /* reset, just in case  */
        sigaction(SIGINT, &act2, NULL);         /* reset, just in case  */
        set_ticker( 500 );

        while(1) {
                ndelay = 0;
                c = read(0, &ch, sizeof(char));
                sprintf(buf, "Read char = %c and errno = %d and EINTR = %d\n",
                        ch, errno, EINTR);
                write(1, buf, strlen(buf));
                if ( c == 'Q' ) break;
                if ( ndelay > 0 )
                        set_ticker( delay = ndelay );
        }
        return 0;
}
```

Επαναλαμβανόμενο ρολόι.
Συνάρτηση στο ticker_demo.c

Μετά από κάθε Ctrl-C πάει στην παρακάτω γραμμή και θέτει το errno.
Αγνοεί το SIGALARM

Παράδειγμα mytest3.c

```
#include        <stdio.h>
#include        <string.h>
#include        <curses.h>
#include        <signal.h>
#include        <unistd.h>
#include        <errno.h>
char buf[128];

void move_msg(int signum)
{
        signal(SIGALRM, SIG_IGN);       /* reset, just in case  */
        signal(SIGINT, move_msg);       /* reset, just in case  */
        strcpy(buf, "Received Alarm\n");
        write(1, buf, strlen(buf));
}

int main() {
        int delay, ndelay, c;
        char ch;

        signal(SIGALRM, SIG_IGN);       /* reset, just in case  */
        signal(SIGINT, move_msg);       /* reset, just in case  */
        set_ticker( 500 );

        while(1) {
                ndelay = 0;
                c = getch();
                sprintf(buf, "Read char = %c and errno = %d and EINTR = %d\n",
                        ch, errno, EINTR);
                write(1, buf, strlen(buf));
                if ( c == 'Q' ) break;
                if ( ndelay > 0 )
                        set_ticker( delay = ndelay );
        }
        return 0;
}
```

Επαναλαμβανόμενο ρολόι.
Συνάρτηση στο ticker_demo.c

Μετά από κάθε Ctrl-C ή σήμα SIGALARM πάει στην παρακάτω γραμμή

ΝΑ ΧΡΗΣΙΜΟΠΟΙΕΙΤΕ ΠΑΝΤΑ read και sigaction

Άλλες συναρτήσεις…

int sigsuspend(const sigset_t *sigmask)

Σαν την pause(), αλλά καθορίζει ότι μπορεί να ξυπνήσει μόνο από σήμα που ΔΕΝ ΕΙΝΑΙ στη sigmask.

int sigwait(const sigset_t *restrict sigmask, int *restrict signo)

Σαν την pause, αλλά περιμένει μόνο για σήματα που ΕΊΝΑΙ στην μάσκα sigmask. Επιστρέφει στο signo το σήμα από το οποίο “ξύπνησε”

Valgrind

Εγκατεστημένο στο
/home/users1/spro_project1/valgrind

Βάλτε το bin κατάλογο στο PATH σας (στο .cshrc για μόνιμα)

setenv PATH
${PATH}:/home/users1/spro_project1/valgrind/bin

Πάντα μεταγλώττιση με –g

Κλήση με:

valgrind --db-attach=yes execFile

Αρχείο Όλο Λάθη

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3
 4 struct koko {
 5   int first;
 6   char second;
 7 };
 8
 9 int main() {
10   struct koko lala1;
11   struct koko *lala2;
12   int *ptr;
13   int a, b;
14
15   a = b;
16
17   a = lala1.first;
18
19   a = lala2->first;
20
21   lala2 = (struct koko *)malloc(sizeof(struct koko));
22
23   ptr[2] = 5;
24
25   return 1;
26 }
```

Valgrind Παράδειγμα

```
linux05:/home/users/spro>gcc -g -o fullOfErrors fullOfErrors.c
linux05:/home/users/spro>valgrind --db-attach=yes ./fullOfErrors
==721== Memcheck, a memory error detector.
==721== Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
==721== Using LibVEX rev 1732, a library for dynamic binary translation.
==721== Copyright (C) 2004-2007, and GNU GPL'd, by OpenWorks LLP.
==721== Using valgrind-3.2.3, a dynamic binary instrumentation framework.
==721== Copyright (C) 2000-2007, and GNU GPL'd, by Julian Seward et al.
==721== For more details, rerun with: -v
==721==
==721== Use of uninitialised value of size 4
==721==    at 0x804838B: main (fullOfErrors.c:19)
==721==
==721== ---- Attach to debugger ? --- [Return/N/n/Y/y/C/c] ---- n
==721==
==721== Use of uninitialised value of size 4
==721==    at 0x80483A5: main (fullOfErrors.c:23)
==721==
==721== ---- Attach to debugger ? --- [Return/N/n/Y/y/C/c] ---- n
==721==
==721== Jump to the invalid address stated on the next line
==721==    at 0x5: ???
==721==    by 0x413AC41: __libc_freeres (in /lib/tls/i686/cmov/libc-2.3.6.so)
==721==    by 0x40192E3: _vgnU_freeres (vg_preloaded.c:60)
==721==    by 0x40BE4F3: _Exit (in /lib/tls/i686/cmov/libc-2.3.6.so)
==721==    by 0x4046EAB: (below main) (in /lib/tls/i686/cmov/libc-2.3.6.so)
==721==  Address 0x5 is not stack'd, malloc'd or (recently) free'd
==721==
==721== ---- Attach to debugger ? --- [Return/N/n/Y/y/C/c] ---- n
==721==
==721== Process terminating with default action of signal 11 (SIGSEGV)
==721==  Bad permissions for mapped region at address 0x5
==721==    at 0x5: ???
==721==    by 0x413AC41: __libc_freeres (in /lib/tls/i686/cmov/libc-2.3.6.so)
==721==    by 0x40192E3: _vgnU_freeres (vg_preloaded.c:60)
==721==    by 0x40BE4F3: _Exit (in /lib/tls/i686/cmov/libc-2.3.6.so)
==721==    by 0x4046EAB: (below main) (in /lib/tls/i686/cmov/libc-2.3.6.so)
==721==
==721== ---- Attach to debugger ? --- [Return/N/n/Y/y/C/c] ---- n
==721==
==721== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 14 from 1)
==721== malloc/free: in use at exit: 8 bytes in 1 blocks.
==721== malloc/free: 1 allocs, 0 frees, 8 bytes allocated.
==721== For counts of detected errors, rerun with: -v
==721== searching for pointers to 1 not-freed blocks.
==721== checked 79,248 bytes.
==721==
==721== LEAK SUMMARY:
```

-g

Valgrind και gdb

Σε κάθε σφάλμα ερώτημα αν θέλετε να συνδεθείτε στο gdb

Αν συνδεθείτε:

bt	Δίνει στοίβα κλήσης εντολών μέχρι το πρόβλημα
list	Λίστα με τις κοντινές εντολές γύρω από το πρόβλημα
up/down	Μετακίνηση πάνω/κάτω στη στοίβα
print	Τυπώνει τη μεταβλητή που του δίνουμε σαν όρισμα
quit	Εγκατάλειψη gdb

Για όσους επιθυμούν εγκατάσταση στο σπίτι

http://valgrind.org/downloads/current.html

- Υπάρχει και γραφικό interface (Valkyrie)
- Κατέβασμα συμπιεσμένου αρχείου valgrind-xxx.tar.bz2
- bunzip2 valgrind-xxx.tar.bz2
- tar xvf valgrind-xxx.tar
- cd valgrind-xxx/

./configure --prefix=$HOME/valgrind
--exec-prefix=$HOME/valgrind

- make
- make install
- Βάλτε το bin κατάλογο του $HOME/valgrind στο PATH
- Τεστ με valgrind ls -l

Σχεδίαση στην οθόνη

Πολύ εύκολο με βιβλιοθήκη curses.h

initscr()	Αρχικοποίηση βιβλιοθήκης και οθόνης
endwin()	Απενεργοποίηση curses και επαναφορά οθόνης
refresh()	Η οθόνη δείχνει τελευταίες αλλαγές
move(r,c)	Ο δρομέας (cursor) μετακινείται στο (r,c)
addstr(s)	Σχεδίασε το string s στην τρέχουσα θέση της οθόνης
addch(c)	Σχεδίασε τον χαρακτήρα c στην τρέχουσα θέση της οθόνης
clear()	Καθάρισμα οθόνης
standout()	Ενεργοποίηση τρόπου standout
standend()	Απενεργοποίηση τρόπου standout

Παράδειγμα – hello1.c

```
/* hello1.c
 *      purpose  show the minimal calls needed to use curses
 *      outline  initialize, draw stuff, wait for input, quit
 */

#include        <stdio.h>
#include        <curses.h>

main()
{
        initscr() ;                 /* turn on curses       */

                                    /* send requests        */
        clear();                            /* clear screen */
        move(10,20);                        /* row10,col20  */
        addstr("Hello, world");             /* add a string */
        move(LINES-1,0);                    /* move to LL   */

        refresh();                  /* update the screen    */
        getch();                    /* wait for user input  */

        endwin();                   /* turn off curses      */
}
```

Παράδειγμα – hello2.c

```
/* hello2.c
 *      purpose  show how to use curses functions with a loop
 *      outline  initialize, draw stuff, wrap up
 */

#include        <stdio.h>
#include        <curses.h>

main()
{
        int     i;

        initscr();                          /* turn on curses       */
           clear();                         /* draw some stuff      */
           for(i=0; i<LINES; i++ ){                 /* in a loop    */
                move( i, i+i );
                if ( i%2 == 1 )
                        standout();
                addstr("Hello, world");
                if ( i%2 == 1 )
                       standend();
           }

           refresh();                       /* update the screen    */
           getch();                         /* wait for user input  */
        endwin();                           /* reset the tty etc    */
}
```

Παράδειγμα – hello3.c

```
/* hello3.c
 *      purpose  using refresh and sleep for animated effects
 *      outline  initialize, draw stuff, wrap up
 */
#include        <stdio.h>
#include        <curses.h>

main()
{
        int     i;

        initscr();
           clear();
           for(i=0; i<LINES; i++ ){
                move( i, i+i );
                if ( i%2 == 1 )
                        standout();
                addstr("Hello, world");
                if ( i%2 == 1 )
                        standend();
                sleep(1);
                refresh();
           }
        endwin();
}
```

Παράδειγμα – hello4.c

```
/* hello4.c
 *      purpose show how to use erase, time, and draw for animation
 */
#include        <stdio.h>
#include        <curses.h>

main()
{
        int     i;

        initscr();
           clear();
           for(i=0; i<LINES; i++ ){
                move( i, i+i );
                if ( i%2 == 1 )
                        standout();
                addstr("Hello, world");
                if ( i%2 == 1 )
                        standend();
                refresh();
                sleep(1);
                move(i,i+i);                    /* move back    */
                addstr("              ");       /* erase line   */
           }
        endwin();
}
```

Παράδειγμα – hello5.c

```
/* hello5.c
 *    purpose  bounce a message back and forth across the screen
 *    compile  cc hello5.c -lcurses -o hello5
 */
#include        <curses.h>

#define LEFTEDGE        10
#define RIGHTEDGE       30
#define ROW             10

main()
{
        char    message[] = "Hello";
        char    blank[]   = "     ";
        int     dir = +1;
        int     pos = LEFTEDGE ;

        initscr();
          clear();
          while(1){
                move(ROW,pos);
                addstr( message );              /* draw string
                move(LINES-1,COLS-1);           /* park the cursor
                refresh();                      /* show string
                sleep(1);
                move(ROW,pos);                  /* erase string
                addstr( blank );
                pos += dir;                     /* advance position
                if ( pos >= RIGHTEDGE )         /* check for bounce
                        dir = -1;
                if ( pos <= LEFTEDGE )
                        dir = +1;
          }
}
```

Παράδειγμα – sleep1.c

```
/* sleep1.c
 *      purpose show how sleep works
 *      usage   sleep1
 *      outline sets handler, sets alarm, pauses, then returns
 */
#include        <stdio.h>
#include        <signal.h>
/* #define SHHHH */

main()
{
        void    wakeup(int);

        printf("about to sleep for 4 seconds\n");
        signal(SIGALRM, wakeup);                /* catch it     */
        alarm(4);                               /* set clock    */
        pause();                                /* freeze here  */
        printf("Morning so soon?\n");           /* back to work */
}

void wakeup(int signum)
{
#ifndef SHHHH
        printf("Alarm received from kernel\n");
#endif
}
```

Να χρησιμοποιείτε sigaction

Παράδειγμα - ticker_demo.c

```
/* ticker_demo.c
 *    demonstrates use of interval timer to generate regular
 *    signals, which are in turn caught and used to count down */
#include        <stdio.h>
#include        <stdlib.h>
#include        <sys/time.h>
#include        <signal.h>

int main() {
        void    countdown(int);

        signal(SIGALRM, countdown);
        if ( set_ticker(500) == -1 )
                perror("set_ticker");
        else
                while( 1 )
                        pause();
        return 0;
}

void countdown(int signum) {
        static int num = 10;
        printf("%d ..", num--);
        fflush(stdout);
        if ( num < 0 ){
                printf("DONE!\n");
                exit(0);
        }
}

/* [from set_ticker.c]
 * set_ticker( number_of_milliseconds )
 *      arranges for interval timer to issue SIGALRM's at regular intervals
 *      returns -1 on error, 0 for ok
 *      arg in milliseconds, converted into whole seconds and microseconds
 *      note: set_ticker(0) turns off ticker
 */
int set_ticker( int n_msecs ) {
        struct itimerval new_timeset;
        long    n_sec, n_usecs;

        n_sec = n_msecs / 1000 ;                /* int part     */
        n_usecs = ( n_msecs % 1000 ) * 1000L ;  /* remainder    */

        new_timeset.it_interval.tv_sec  = n_sec;        /* set reload       */
        new_timeset.it_interval.tv_usec = n_usecs;      /* new ticker value */
        new_timeset.it_value.tv_sec     = n_sec  ;      /* store this       */
        new_timeset.it_value.tv_usec    = n_usecs ;     /* and this         */

        return setitimer(ITIMER_REAL, &new_timeset, NULL);
}
```

Πότε να χτυπήσει πρώτη φορά και μετά κάθε πόσο

Παράδειγμα – bounce1d.c

```
/* bounce1d.c
 *      purpose animation with user controlled speed and direction
 *      note    the handler does the animation
 *              the main program reads keyboard input
 *      compile cc bounce1d.c set_ticker.c -lcurses -o bounce1d
 */
#include        <stdio.h>
#include        <string.h>
#include        <curses.h>
#include        <signal.h>

/* some global settings main and the handler use */
#define MESSAGE "hello"
#define BLANK   "     "

int     row;    /* current row          */
int     col;    /* current column       */
int     dir;    /* where we are going   */

int main() {
        int     delay;          /* bigger => slower     */
        int     ndelay;         /* new delay            */
        int     c;              /* user input           */
        void    move_msg(int);  /* handler for timer    */

        initscr();
        crmode();
        noecho();
        clear();

        row   = 10;             /* start here           */
        col   = 0;
        dir   = 1;              /* add 1 to row number  */
        delay = 200;            /* 200ms = 0.2 seconds  */

        move(row,col);          /* get into position    */
        addstr(MESSAGE);        /* draw message         */
        signal(SIGALRM, move_msg );
        set_ticker( delay );

        while(1) {
                ndelay = 0;
                c = getch();
                if ( c == 'Q' ) break;
                if ( c == ' ' ) dir = -dir;
                if ( c == 'f' && delay > 2 ) ndelay = delay/2;
                if ( c == 's' ) ndelay = delay * 2 ;
                if ( ndelay > 0 )
                        set_ticker( delay = ndelay );
        }
        endwin();
        return 0;
}
```

Παράδειγμα – bounce1d.c

```
void move_msg(int signum)
{
        signal(SIGALRM, move_msg);      /* reset, just in case  */
        move( row, col );
        addstr( BLANK );
        col += dir;                     /* move to new column   */
        move( row, col );               /* then set cursor      */
        addstr( MESSAGE );              /* redo message         */
        refresh();                      /* and show it          */

        /*
         * now handle borders
         */
        if ( dir == -1 && col <= 0 )
                dir = 1;
        else if ( dir == 1 && col+strlen(MESSAGE) >= COLS )
                dir = -1;
}
```

Παράδειγμα – bounce2d.c

- Αυτό κοιτάξτε το μόνοι σας…
- Δύο συχνότητες ανανέωσης
  - Κίνηση σε Χ διάσταση
  - Κίνηση σε Υ διάσταση
  - Τι γίνεται στο παιχνίδι αν αυξήσουμε πολύ την ταχύτητα σε μία διάσταση? Γιατί?