ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ
Εξετάσεις Α΄ Περιόδου 2007 (3/7/2007)

1. (α΄) Παρατηρήστε την παρακάτω διαλογική επικοινωνία με το κέλυφος και συμπληρώστε τα αποτελέσματα στα αποσιωπητικά. Δικαιολογήστε σε **15 γραμμές το πολύ** την απάντησή σας.

```
$ ls -l *
-rwx------ 1 syspro users 341 Jun 29 20:59 quotes
$ cat quotes
#!/bin/sh
#
echo $1         # $1
echo '$1'       # single $1 single
echo "$1"       # double $1 double
echo '"$1"'     # single double $1 double single
echo "'$1'"     # double single $1 single double
echo '"'$1'"'   # single double single $1 single double single
echo '"'"$1"'"' # single double single double $1 double single double single
$ ./quotes "abc * de"
.....................................................
$ ./quotes "  abc   * de"
.....................................................
```

   (β΄) Γνωρίζετε ότι στην επικοινωνία μεταξύ διεργασιών μέσω σωλήνων ή υποδοχών ροής δεν υποστηρίζονται από το ίδιο το πρωτόκολλο όρια μεταξύ των μεταδιδόμενων μηνυμάτων. Προτείνετε επιγραμματικά (σε **6 γραμμές το πολύ**) τρεις τρόπους για να υλοποιήσει κανείς όρια μεταξύ μηνυμάτων, σε μία εφαρμογή που χρησιμοποιεί αυτούς τους τρόπους επικοινωνίας. Γράψτε, για έναν από αυτούς τους τρόπους, και μόνο για την παραλαβή μηνυμάτων, τμήμα προγράμματος C που εγγυάται την ανάγνωση ενός αυτοτελούς μηνύματος.

2. Γράψτε ένα πρόγραμμα κελύφους Bourne (έστω ότι ονομάζεται "**procmarks**") που να συμβουλεύεται ένα αρχείο δεδομένων, το όνομα-μονοπάτι του οποίου είναι η τιμή της μεταβλητής περιβάλλοντος `MARKSFILE`, στο οποίο περιέχονται οι τελικοί βαθμοί ενός μαθήματος πρώτου έτους σ΄ ένα πανεπιστημιακό τμήμα. Το αρχείο δεδομένων είναι ένα απλό αρχείο κειμένου, κάθε γραμμή του οποίου περιέχει τους βαθμούς των φοιτητών ενός έτους (ή πρώτη γραμμή του πρώτου έτους, η δεύτερη του δεύτερου, κοκ.) που έχουν δηλώσει το μάθημα. Αν ένας φοιτητής δεν προσήλθε στις εξετάσεις, τότε στη θέση του βαθμού του υπάρχει το σύμβολο `#`. Υποθέστε ότι το αρχείο δίνεται στη σωστή του μορφή. Το πρόγραμμα "**procmarks**" που θα γράψετε να διαβάζει τα δεδομένα από το αρχείο και να εκτυπώνει, για κάθε έτος, αλλά και συνολικά, το ποσοστό των φοιτητών που προσήλθαν στις εξετάσεις, το ποσοστό των φοιτητών που πέτυχαν στο μάθημα (βαθμός ≥ 5), σε σχέση με αυτούς που προσήλθαν στις εξετάσεις, και τον μέσο βαθμό των φοιτητών. Τα ποσοστά να δίνονται σαν ακέραιοι (με αποκοπή του δεκαδικού μέρους) και οι μέσοι βαθμοί με ένα δεκαδικό ψηφίο (με αποκοπή των υπόλοιπων δεκαδικών). Μία ενδεικτική εκτέλεση είναι η εξής:

```
$ cat marks.txt
9 8 7 # # 8 10 8 6 # 3 2 8 2 0 8 9 # 8
9 # # 10 4 9 8 6 7 8 9 5 # 4 9 7
# # 9 7 9 0 8 4 6 10
7 8 6 9 # 2 8 7
1 6 9 5 4 8 # # #
$ MARKSFILE=marks.txt ; export MARKSFILE
$ ./procmarks
```

```
Year 1: Took the exam = 78%  Passed = 73%  Average mark = 6.4
Year 2: Took the exam = 81%  Passed = 84%  Average mark = 7.3
Year 3: Took the exam = 80%  Passed = 75%  Average mark = 6.6
Year 4: Took the exam = 87%  Passed = 85%  Average mark = 6.7
Year 5: Took the exam = 66%  Passed = 66%  Average mark = 5.5
Total:  Took the exam = 79%  Passed = 77%  Average mark = 6.6
```

3. Μελετήστε το τμήμα προγράμματος C στο διπλανό σχήμα και εξηγήστε πολύ συνοπτικά τη λειτουργία του. Αν το εκτελέσιμο πρόγραμμα που προκύπτει απ΄ αυτό ονομάζεται "**hide**", συμπληρώστε τα αποτελέσματα των εντολών που φαίνονται στη συνέχεια. Υποθέστε ότι όλες οι κλήσεις συστήματος θα είναι επιτυχείς.

```
$ wc -c hide.c
616 hide.c
$ cat inp.txt
abcdefghijklmno
ABCDEFGHIJKLMNO
123456789012345
678901234567890
$ ./hide 1 < inp.txt
.............................
$ ./hide 3 < inp.txt
.............................
$ ./hide 8 < hide.c | wc -c
.............................
```

```
.....................................
main(int argc, char *argv[])
{  int i, n, status, fd[2];
   int flag = 1; char ch;
   n = atoi(argv[1]);
   for (i=0 ; i<n ; i++) {
      if (i != n-1) pipe(fd);
      if (fork() == 0) {
         if (i != n-1) {
            close(fd[0]);
            dup2(fd[1], 1);
            close(fd[1]); }
         while (read(0, &ch, 1) > 0) {
            if (flag < 0) write(1, &ch, 1);
            flag = -flag; }
         exit(0); }
      else {
         if (i != n-1) {
            dup2(fd[0], 0);
            close(fd[0]);
            close(fd[1]); } } }
.....................................
```

4. Γράψτε ένα πρόγραμμα C (έστω ότι ονομάζεται "**hardlinks**") το οποίο, όταν καλείται σαν **hardlinks** $\langle node \rangle$ [$\langle dir \rangle_1$ $\langle dir \rangle_2$ ... $\langle dir \rangle_n$], να βρίσκει πόσοι σκληροί σύνδεσμοι υπάρχουν για το $\langle node \rangle$ (όνομα κόμβου στο σύστημα αρχείων, δηλαδή απλό αρχείο, κατάλογος, κλπ.) και μετά να ελέγχει αν κάποιοι απ΄ αυτούς βρίσκονται κάτω από τους καταλόγους $\langle dir \rangle_1$ $\langle dir \rangle_2$ ... $\langle dir \rangle_n$. Οι κατάλογοι είναι προαιρετικοί. Αν δεν δοθούν, ο έλεγχος να γίνει κάτω από τον τρέχοντα κατάλογο. Κάποιες ενδεικτικές εκτελέσεις:

```
$ ln hardlinks.c ..
$ ln hardlinks.c mydir/myfile.c
$ ./hardlinks hardlinks.c ../sp_programs .. mydir
There are 3 hard link(s) for hardlinks.c
Found hardlink at ../progs/hardlinks.c
Found hardlink at ../progs/mydir/myfile.c
Found hardlink at ../hardlinks.c
Found hardlink at mydir/myfile.c
$ ./hardlinks . ..
There are 3 hard link(s) for .
Found hardlink at ../progs
Found hardlink at ../progs/.
Found hardlink at ../progs/mydir/..
$ ./hardlinks ~
There are 25 hard link(s) for /home/users/syspro
Found hardlink at ./..
```