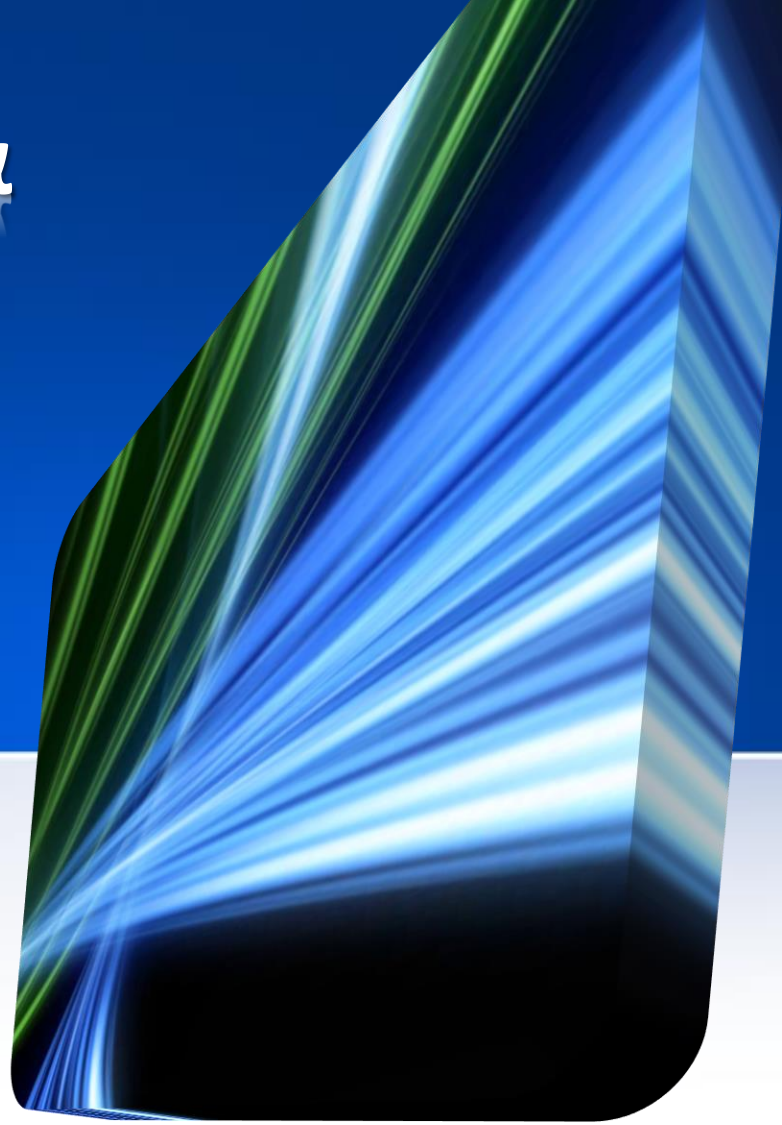
Λειτουργικά Συστήματα
6ο εξάμηνο ΣΗΜΜΥ
Ακ. έτος 2020-2021



Εργαστηριακή Άσκηση 1

Κανονισμός εργαστηριακών ασκήσεων

- Οι εργαστηριακές ασκήσεις διεξάγονται σε δύο Σειρές
  Σειρά Α (Τρίτη 12.45 – 14.45) και
  Σειρά Β (Τρίτη 15.00 – 17.00)

- Οι φοιτητές είναι χωρισμένοι σε ομάδες των 2 ατόμων με αρίθμηση Σειρά Α1 – Σειρά Α35 και Σειρά Β1 – Σειρά Β35

Κανονισμός εργαστηριακών ασκήσεων

- Θα εκτελεστούν 4 εργαστηριακές ασκήσεις με βαρύτητα 10%, 30%, 30% και 30% επί του τελικού βαθμού εργαστηρίου

- Κάθε ομάδα πρέπει να προσέλθει για την προφορική εξέταση της κάθε άσκησης σε συγκεκριμένη ημερομηνία και ώρα

Κανονισμός εργαστηριακών ασκήσεων

- Δεν υπάρχουν υποχρεωτικές παρουσίες στο εργαστήριο εκτός από τις ημερομηνίες εξέτασης των ασκήσεων
- Δεν απαιτείται παράδοση γραπτών αναφορών των εργαστηριακών ασκήσεων

Κανονισμός εργαστηριακών ασκήσεων

- Κάθε ομάδα μπορεί προαιρετικά να προσέλθει στο εργαστήριο κάθε εβδομάδα την ώρα του Τμήματος της για διατύπωση απορίων ή διευκρινήσεων, πρέπει όμως να εξετάζεται στις προκαθορισμένες ημερομηνίες και ώρες.

Εργαστηριακή Άσκηση 1

Να γραφτεί πρόγραμμα σε γλώσσα προγραμματισμού C και περιβάλλον Linux το οποίο δέχεται δύο ορίσματα από την γραμμή εντολών. Το πρώτο όρισμα είναι το όνομα ενός αρχείου (**filename**) και το δεύτερο όρισμα είναι ένας ακέραιος (**n**).

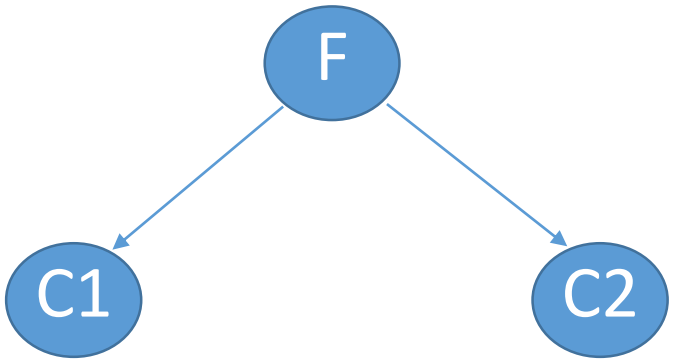
Παράδειγμα εκτέλεσης:

```
./ask1 a.txt 15
```

filename n

Εργαστηριακή Άσκηση 1

Η διεργασία πατέρας (F) δημιουργεί 2 διεργασίες (C1, C2) σύμφωνα με το παρακάτω δέντρο διεργασιών



Εργαστηριακή Άσκηση 1

Κάθε διεργασία παιδί (C1, C2)

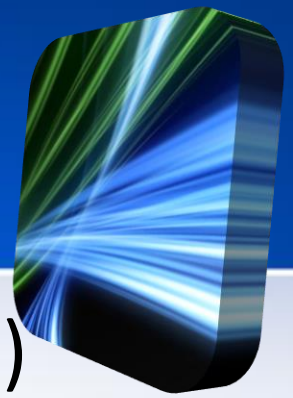
εκτυπώνει αρχικά το παρακάτω μήνυμα

[Childi] Started. PID= Pid PPID=Ppid

όπου Pid είναι το **process id** της διεργασίας

και Ppid είναι το **process id** της διεργασίας που την έχει δημιουργήσει, ενώ i είναι η αρίθμηση της δλδ 1 ή 2

Εργαστηριακή Άσκηση 1



Στην συνέχεια κάθε διεργασία παιδί (C1, C2) εκτυπώνει το παρακάτω μήνυμα επαναληπτικά

[Childi] Heartbeat PID= Pid Time=time x=k

Όπου k είναι ένα αριθμός μεταξύ **0** και **n**.

Το Child1 εκτυπώνει (k) επαναληπτικά τους ζυγούς αριθμούς έως το **n**.

Το Child2 εκτυπώνει (k) επαναληπτικά τους μονούς αριθμούς έως το **n**.

time είναι η χρονική στιγμή της εκτύπωσης που θα την βρείτε με χρήση της **time()**

Εργαστηριακή Άσκηση 1

Σε κάθε επανάληψη κάθε διεργασία παιδί (C1, C2) γράφει στο αρχείο το παρακάτω μήνυμα

message from Pid

Εργαστηριακή Άσκηση 1

- Η διεργασία πατέρας δημιουργεί αρχικά το αρχείο με όνομα **filename** και δημιουργεί τις 2 διεργασίες με χρήση της κλήσης συστήματος **fork()**
- Στην συνέχεια και αυτός εκτυπώνει επαναληπτικά το παρακάτω μήνυμα

  [Parent] Heartbeat PID= Pid Time=time
- Έπειτα περιμένει τις διεργασίες παιδιά να τερματίσουν και διαβάζει και εκτυπώνει το περιεχόμενο του αρχείου.

Εργαστηριακή Άσκηση 1

Προσδοκώμενο αποτέλεσμα:

Δεδομένου ότι οι διεργασίες εκτελούνται ταυτόχρονα και χρονοδρομολογούνται από το Λειτουργικό Σύστημα, τα μηνύματα θα πρέπει να εμφανίζονται ανακατεμένα, και όχι όλα τα μηνύματα κάθε διεργασίας μαζεμένα.

Εργαστηριακή Άσκηση 1

Προσδοκώμενο αποτέλεσμα:

```
root@LAPTOP-89VKCAQQ:~# gcc ask1.c -o ask1
root@LAPTOP-89VKCAQQ:~# ./ask1 a.txt 15
[Child1] Started. PID=81 PPID=80
[Child1] Heartbeat PID=81 Time=1614674169 x=0
[Parent] Heartbeat PID=80 Time=1614674169
[Child2] Started. PID=82 PPID=80
[Child2] Heartbeat PID=82 Time=1614674169 x=1
[Child1] Heartbeat PID=81 Time=1614674170 x=2
[Parent] Heartbeat PID=80 Time=1614674170
…
[Parent] Waiting for child
[Child2] Terminating!
[Parent] Child with PID=82 terminated
[Parent] Waiting for child
[Child1] Terminating!
[Parent] Child with PID=81 terminated
[Parent] PID=80 Reading file:
message from 81
message from 80
message from 82
message from 81
message from 80
message from 82
…
```

Θεωρία Εργ. Άσκησης 1

Διεργασία είναι ένα πρόγραμμα που εκτελείται

Είναι μια μονάδα εργασίας **μέσα** στο σύστημα. Το πρόγραμμα είναι μια παθητική οντότητα, η διεργασία είναι μια **ενεργή οντότητα**.

Η διεργασία χρειάζεται

- **πόρους** (CPU, μνήμη, μονάδες Ε/Ε, αρχεία) για την εκπλήρωση των καθηκόντων της
- **δεδομένα** αρχικοποίησης

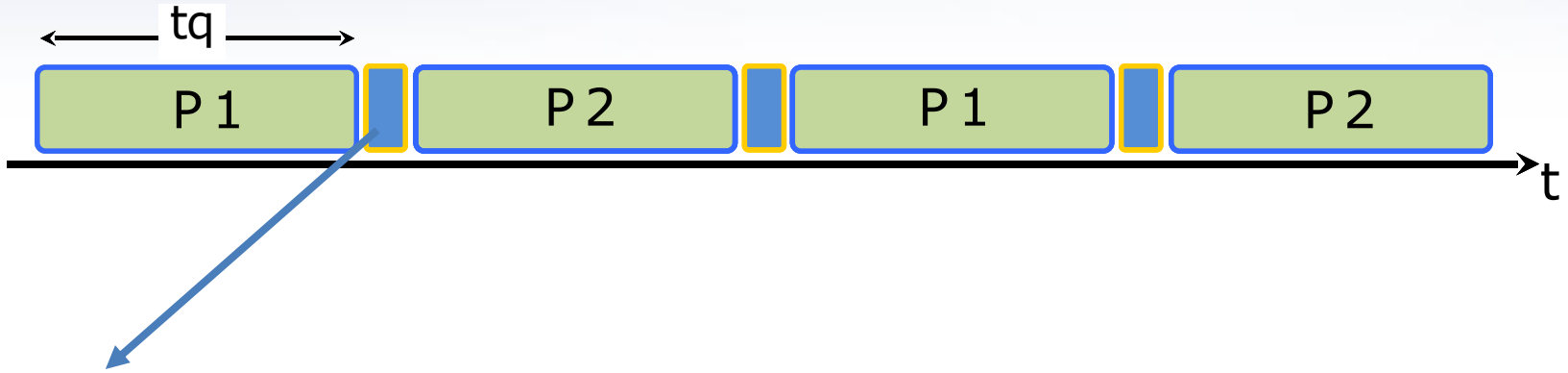
Θεωρία Εργ. Άσκησης 1

Μοντέλο Διαμοιρασμού Χρόνου

- Πολλαπλές διεργασίες θέλουν να εκτελεστούν ταυτόχρονα.
- Το Λειτουργικό Σύστημα μοιράζει τον χρόνο του επεξεργαστή και αναλαμβάνει να τις χρονοδρομολογίσει.
- Οι διεργασίες έχουν την (ψευδ)αίσθηση ότι χρησιμοποιούν αποκλειστικά τον επεξεργαστή
- Ο χρονοδρομολογητής αναλαμβάνει:
    - Την επιλογή της διεργασίας που θα χρησιμοποιήσει τον επεξεργαστή
    - Την αλλαγή της διεργασίας που εκτελείται στον επεξεργαστή (context switch)

Θεωρία Εργ. Άσκησης 1

Μοντέλο Διαμοιρασμού Χρόνου



- Επιλογή επόμενης διεργασίας (scheduling)
- Αλλαγή περιβάλλοντος λειτουργίας (Context Switch)

Χρήσιμες συναρτήσεις

Κάθε διεργασία έχει συσχετισμένο μαζί της έναν εγγυημένα μοναδικό αριθμό ταυτότητας διεργασίας(process-id, pid) που παρέχεται δυναμικά από το Λειτουργικό Σύστημα. Ο αριθμός αυτός χρησιμοποιείται για να αναφερθούμε σε κάποια διεργασία.

Μια διεργασία μπορεί να μάθει το pid της εκτελώντας την κλήση:

```
pid_t  getpid(void)
```

Το pid μιας διεργασίας μπορεί να αποθηκευτεί σε μια μεταβλητή τύπου **pid_t**

Χρήσιμες συναρτήσεις

Παράδειγμα 1: Μία διεργασία ενημερώνεται για το pid της και στη συνέχεια το εκτυπώνει.

```
#include<stdio.h>
#include <stdlib.h>
int main()
{
        pid_t mypid;
        mypid = getpid() ;
        printf(" My id: %d\n", mypid);
        return(0);
}
```

Χρήσιμες συναρτήσεις

Κάθε διεργασία μπορεί να μάθει το αριθμό ταυτότητας (pid) της γονικής διεργασίας (δηλαδή της διεργασίας που τη δημιούργησε) χρησιμοποιώντας την εντολή getppid() εκτελώντας την κλήση:

```
pid_t getppid(void)
```

Χρήσιμες συναρτήσεις

Παράδειγμα 2: Μία διεργασία ενημερώνεται για το pid της γονικής διεργασίας και στη συνέχεια το εκτυπώνει.

```
pid_t parent_pid;
parent_pid = getppid() ;
printf(“ My parent’s id: %d\n”, parent_pid);
```

Χρήσιμες συναρτήσεις

Μια διεργασία μπορεί να δημιουργήσει μια νέα διεργασία-παιδί, πιστό αντίγραφο του εαυτού της με χρήση της κλήσης fork()

Η κλήση fork() επιστρέφει την τιμή 0 στην διεργασία παιδί και το pid του παιδιού στην διεργασία πατέρα.

Με τον τρόπο αυτό η διεργασία-παιδί που προέκυψε μπορεί να αντικαθιστά το πρόγραμμα που εκτελεί (αρχικά ίδιο με του πατέρα) με νέο πρόγραμμα.

Χρήσιμες συναρτήσεις

Ο γονέας μπορεί να περιμένει μέχρι τον τερματισμό κάποιας διεργασίας-παιδιού του με την κλήση wait() .

Η κλήση wait() αναστέλλει την εκτέλεση του καλούντος προγράμματος μέχρις ότου τερματισθεί η εκτέλεση κάποιας από τις διεργασίες παιδιά του. Η συνάρτηση wait() επιστρέφει το pid της θυγατρικής διεργασίας ή -1 για σφάλμα. Η κατάσταση εξόδου της θυγατρικής διεργασίας βρίσκεται στη μεταβλητή status. Επίσης, αν κάποια διεργασία παιδί έχει ήδη τερματιστεί, τότε η κλήση επιστρέφει αμέσως -1.

Ο οικειοθελής τερματισμός μιας διεργασίας μπορεί να γίνει με τη κλήση exit()

Χρήσιμες συναρτήσεις

Παράδειγμα 3: Μία διεργασία δημιουργεί μια νέα διεργασία

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
        int status;
        pid_t child ;
        child = fork();
        if(child<0){
                //error
                ...
        }
        if(child==0){
                //child's code
                ...
                exit(0);
        }
        else {
                //father's code
                ...
                wait(&status);
                exit(0);
        }
    return 0;
}
```

Χρήσιμες συναρτήσεις

Η εντολή sleep() εισάγει μια αναμονή στο σύστημα, για όσα δευτερόλεπτα της δώσουμε ως παράμετρο.

Παράδειγμα 4. Αναμονή 5 δευτερολέπτων

```
sleep(5)
```

Παράρτημα

Ακολουθούν χρήσιμες πληροφορίες χρήσης περιβάλλοντος προγραμματισμού

Χρήσιμες πληροφορίες χρήσης περιβάλλοντος προγραμματισμού

Μονοπάτι (path):
Συμβολοσειρα από αναγνωριστικά χωρισμένα από τον χαρακτήρα **/**
πχ: `/home/christos/first.c`

Το μονοπάτι είναι

- *απόλυτο* αν ξεκινάει με **/** → αφετηρία είναι η αρχή της ιεραρχίας
- *σχετικό* → αφετηρία είναι ο τρέχων κατάλογος (ΤΚ)

Το αναγνωριστικό:

. σηματοδοτεί τον ΤΚ
.. σηματοδοτεί τον πατέρα του ΤΚ

Διαχείριση καταλόγων

Εντολές

cd: Αλλαγή τρέχοντος καταλόγου

mkdir: Δημιουργία καταλόγου

rmdir: Διαγραφή καταλόγου

Διαχείριση Αρχείων

Εντολές

cat: Εκτύπωση

cp: Αντιγραφή

mv: Μετακίνηση

rm: Διαγραφή

Compiling & linking

- **Compile** (Μεταγλώττιση):

first.c ⇒ first.o

second.c ⇒ second.o

- **Link** (Σύνδεση):

first.o + second.o ⇒ executable

Παράδειγμα compiling & linking ενός αρχείου

```
$ gcc  -Wall  -c   first.c
$ gcc  first.o     -o     first
$ ./first
```

ή

```
$ gcc  -Wall    first.c –o first
$ ./first
```

Παράδειγμα compiling & linking πολλαπλών αρχείων

```
$ gcc -Wall -c first.c
$ gcc -Wall -c second.c

$ gcc first.o second.o –o allinone
$ ./allinone
```

Χρήσιμα Links

https://help.ubuntu.com/community/UsingTheTerminal

https://files.fosswire.com/2007/08/fwunixref.pdf

http://www.gnu.org/software/libc/manual/html_node/Processes.html#Processes