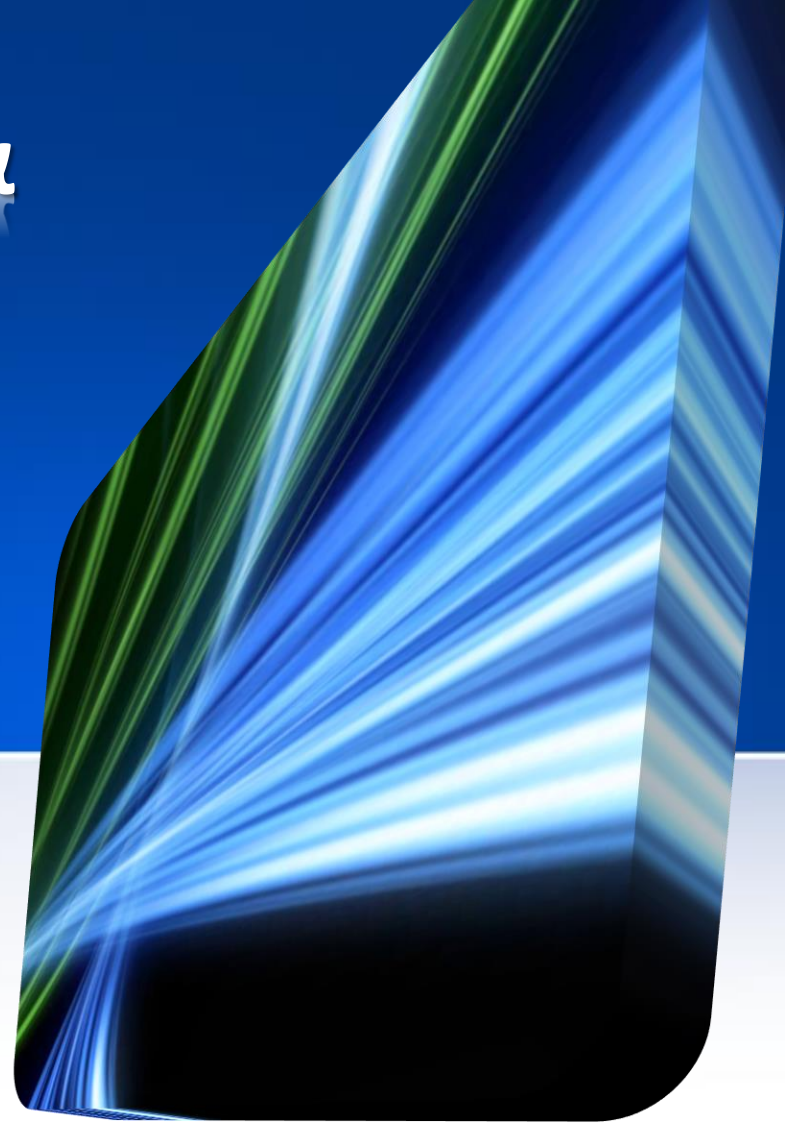


Λειτουργικά Συστήματα 6ο εξάμηνο ΣΗΜΜΥ Ακ. έτος 2018-2019

Εργαστηριακή Άσκηση 1



Κανονισμός εργαστηριακών ασκήσεων



- ❑ Οι εργαστηριακές ασκήσεις διεξάγονται σε δύο τμήματα Τμήμα Α (Τρίτη 10.30 - 12.30) και Τμήμα Β (Τρίτη 12.30 - 14.30)
- ❑ Οι φοιτητές είναι χωρισμένοι σε ομάδες των 2 ατόμων με αρίθμηση Τμήμα Α1 - Τμήμα Α75 και Τμήμα Β1 - Τμήμα Β75

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

Κανονισμός εργαστηριακών ασκήσεων



Ημερομηνίες εξέτασης των εργαστηριακών ασκήσεων

	26/2	5/3	12/3	19/3	26/3	2/4	9/4	16/4	23/4	30/4	7/5	14/5	21/5	28/5
	Εργαστηριακή Άσκηση 1			Εργαστηριακή Άσκηση 2			Εργαστηριακή Άσκηση 3		Πάσχα		Εργ. Άσκηση 3	Εργαστηριακή Άσκηση 4		
10.30 - 12.30		Ομάδα A1- A38	Ομάδα A36- A75	Ομάδα A1- A25	Ομάδα A26- A50	Ομάδα A51- A75	Ομάδα A1- A25	Ομάδα A26- A50			Ομάδα A51- A75	Ομάδα A1- A25	Ομάδα A26- A50	Ομάδα A51- A75
12.30 - 14.30		Ομάδα B1- B38	Ομάδα B36- B75	Ομάδα B1- B25	Ομάδα B26- B50	Ομάδα B51- B75	Ομάδα B1- A25	Ομάδα B26- B50			Ομάδα B51- B75	Ομάδα B1- B25	Ομάδα A26- A50	Ομάδα B51- B75

Κανονισμός εργαστηριακών ασκήσεων



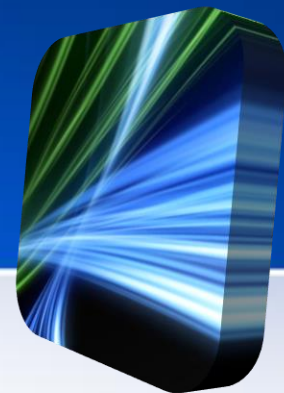
- ☐ Δεν υπάρχουν υποχρεωτικές παρουσίες στο εργαστήριο εκτός από τις ημερομηνίες εξέτασης των ασκήσεων
- ☐ Δεν απαιτείται παράδοση γραπτών αναφορών των εργαστηριακών ασκήσεων

Κανονισμός εργαστηριακών ασκήσεων



- ❑ Κάθε ομάδα μπορεί προαιρετικά να προσέλθει στο εργαστήριο κάθε εβδομάδα την ώρα του Τμήματος της για διατύπωση αποριών ή διευκρινίσεων, πρέπει όμως να εξετάζεται στις προκαθορισμένες ημερομηνίες
- ❑ Κάθε εργαστηριακή άσκηση θα παρουσιάζεται Δευτέρα την ώρα των θεωρητικών διαλέξεων, σύμφωνα με το παρακάτω πρόγραμμα έτσι ώστε όλες οι ομάδες να έχουν αρκετό χρόνο να τις προετοιμάσουν πριν εξεταστούν

Κανονισμός εργαστηριακών ασκήσεων



Ημερομηνίες παρουσίασης των εργαστηριακών ασκήσεων

Εργαστηριακή Άσκηση 1

25/2

Εργαστηριακή Άσκηση 2

4/3

Εργαστηριακή Άσκηση 3

18/3

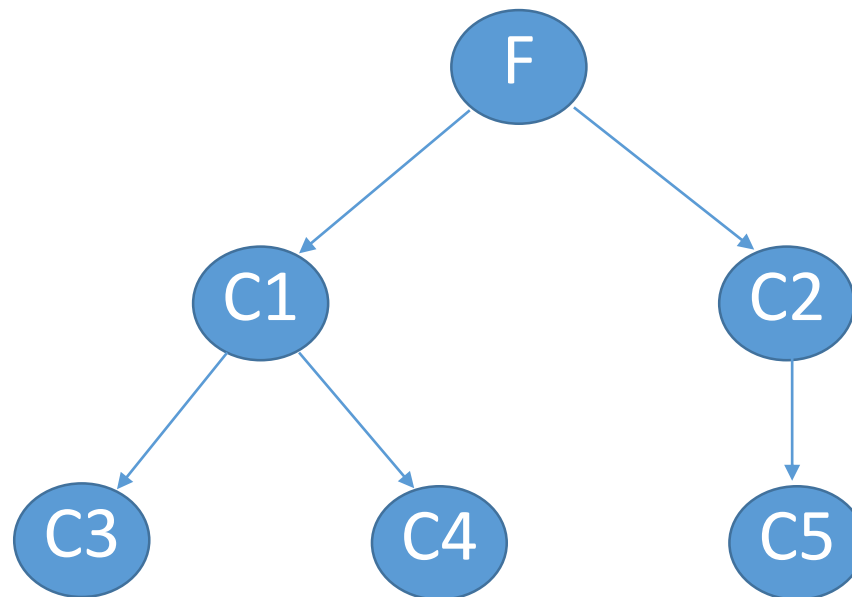
Εργαστηριακή Άσκηση 4

8/4

Εργαστηριακή Άσκηση 1



Να γραφτεί πρόγραμμα σε γλώσσα προγραμματισμού C και περιβάλλον Linux στο οποίο η διεργασία πατέρας (F) δημιουργεί 5 διεργασίες (C1,C2,C3,C4,C5) σύμφωνα με το παρακάτω δέντρο διεργασιών



Εργαστηριακή Άσκηση 1



Κάθε διεργασία παιδί (C1, C2, C3, C4, C5)
εκτυπώνει επαναληπτικά 10 φορές το εξής
μήνυμα με καθυστέρηση ενός δευτερολέπτου:

Process **Pid** is executed, my father is **Ppid**

όπου **Pid** είναι το **process id** της διεργασίας
και **Ppid** είναι το **process id** της διεργασίας που
την έχει δημιουργήσει

Εργαστηριακή Άσκηση 1



Προσδοκώμενο αποτέλεσμα:

Θα εμφανιστούν 50 μηνύματα στην οθόνη.

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

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



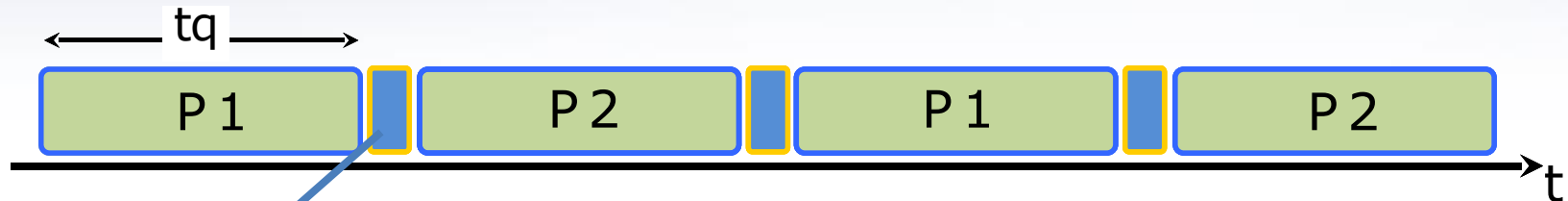
Μοντέλο Διαμοιρασμού Χρόνου

- Πολλαπλές διεργασίες θέλουν να εκτελεστούν ταυτόχρονα.
- Το Λειτουργικό Σύστημα μοιράζει τον χρόνο του επεξεργαστή και αναλαμβάνει να τις χρονοδρομολογήσει.
- 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

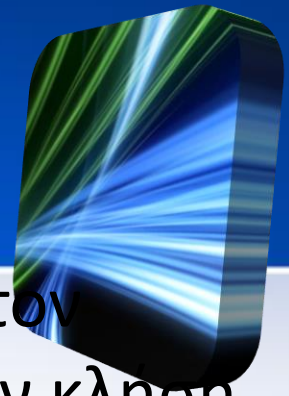


Μια διεργασία μπορεί να δημιουργήσει μια νέα διεργασία-παιδί, πιστό αντίγραφο του εαυτού της με χρήση της κλήσης `fork()`

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



Προαιρετικά, 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)`



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 (TK)

Το αναγνωριστικό:

- σηματοδοτεί τον TK
- .. σηματοδοτεί τον πατέρα του TK

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 \Rightarrow first.o

second.c \Rightarrow second.o

❑ **Link** (Σύνδεση):

first.o + second.o \Rightarrow
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