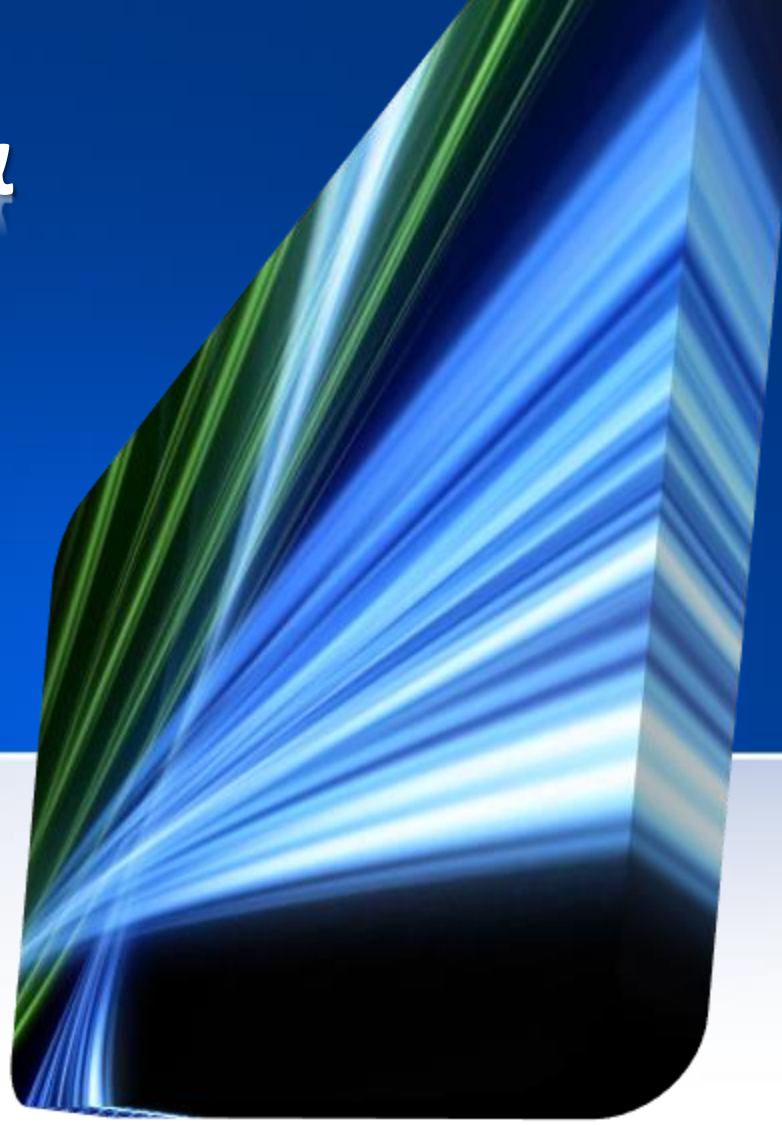
Λειτουργικά Συστήματα
6ο εξάμηνο ΣΗΜΜΥ
Ακ. έτος 2023-2024



Εργαστηριακή Άσκηση 1

Κανονισμός εργαστηριακών ασκήσεων

- Οι εργαστηριακές ασκήσεις διεξάγονται σε δύο Σειρές
  Σειρά Α (Τρίτη 09.00 – 10.45) και
  Σειρά Β (Τρίτη 11.00 – 12.45)

- Οι φοιτητές είναι χωρισμένοι σε ομάδες των 2 ατόμων με αρίθμηση Σειρά Α1 – Σειρά Α30 και Σειρά Β1 – Σειρά Β30

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

- Δεν υπάρχουν υποχρεωτικές παρουσίες στο εργαστήριο εκτός από τις ημερομηνίες εξέτασης των ασκήσεων

- Δεν απαιτείται παράδοση γραπτών αναφορών των εργαστηριακών ασκήσεων

Slack

- Το Slack είναι το επίσημο κανάλι επικοινωνίας για το εργαστήριο
- Αποφύγετε να κάνετε spam στο κανάλι #general.
- Σε απορίες που αφορούν το περιεχόμενο των ασκήσεων: γράφετε την απορία στο #general, μαζί με το τι έχετε δοκιμάσει, μαζί με το μήνυμα σφάλματος.

Slack

- Αν κάποιος άλλος φοιτητής ή ομάδα πιστεύει ότι μπορεί να βοηθήσει, παρακαλείται να **μην** το κάνει. Στις απορίες σας θα απαντούμε εμείς, καλύπτοντας την ερώτηση, και δίνοντας πληροφορίες και συνδέσμους για περισσότερες πληροφορίες.
- Ανακοινώσεις και μηνύματα που γράφονται στο Slack θεωρούνται γνωστά τοις πάσι.

Εργαστηριακή Άσκηση 1

- Εισαγωγή στις εντολές φλοιού σε περιβάλλον Linux
- Εισαγωγή στο περιβάλλον προγραμματισμού
- Διαχείριση αρχείων
- Δημιουργία διεργασιών

Εργαστηριακή Άσκηση 1

Να γραφτεί πρόγραμμα σε γλώσσα προγραμματισμού C και περιβάλλον Linux στο οποίο η διεργασία πατέρας (F) δημιουργεί 1 διεργασία παιδί (C1). Οι διεργασίες F και C1 γράφουν από ένα μήνυμα σ' ένα αρχείο. Το όνομα του αρχείου προσδιορίζεται από τον χρήστη ως όρισμα από την γραμμή εντολών κατά την εκτέλεση του προγράμματος

Εργαστηριακή Άσκηση 1

Κάθε διεργασία παιδί γράφει στο αρχείο το εξής μήνυμα

[CHILD] getpid()= Pid, getppid()=Ppid

[PARENT] getpid()= Pid, getppid()=Ppid

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

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

Εργαστηριακή Άσκηση 1

Παράδειγμα εκτέλεσης:

```
$ gcc lab1.c
$ ./a.out output.txt
```

Προσδοκώμενο αποτέλεσμα:

```
$ cat output.txt
[CHILD] getpid()= 100, getppid()=99
[PARENT] getpid()= 99, getppid()=14
```

Διευκρίνηση 1

Το πρόγραμμα δέχεται από την γραμμή εντολών κατά την εκτέλεσή του ένα και μοναδικό όρισμα, το οποίο είναι ένα όνομα αρχείου. Για παράδειγμα, το πρόγραμμα θα πρέπει να εκτελείται ως εξής:

$ ./a.out output.txt

Αν το πρόγραμμα κληθεί χωρίς κανένα όρισμα, ή με δύο ή περισσότερα ορίσματα, τότε πρέπει να τυπώνει το εξής μήνυμα σφάλματος και να τερματίζει με κωδικό επιστροφής 1:

$ ./a.out
Usage: ./a.out filename

$ ./a.out file1.txt file2.txt
Usage: ./a.out filename

Διευκρίνηση 2

Αν το αρχείο που έχει δοθεί στο πρόγραμμα υπάρχει ήδη, τότε το πρόγραμμα τυπώνει το παρακάτω μήνυμα σφάλματος και τερματίζει με κωδικό επιστροφής 1:

$ ./a.out output.txt

Error: output.txt already exists

HINT: Για τον έλεγχο ύπαρξης του αρχείου, δείτε την κλήση συστήματος ***stat.***

Διευκρίνηση 3

Το πρόγραμμα επίσης πρέπει να μπορεί να κληθεί μαζί με το flag --help, στην οποία περίπτωση τυπώνει το παρακάτω μήνυμα στην έξοδο (stdout) και τερματίζει με κωδικό επιστροφής 0:

```
$ ./a.out --help
Usage: ./a.out filename
```

Διευκρίνηση 4

Για την εγγραφή στο αρχείο **πρέπει** να χρησιμοποιήσετε την κλήση συστήματος ***write***.

Διευκρίνηση 5

Να γίνουν κατάλληλοι έλεγχοι λαθών

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

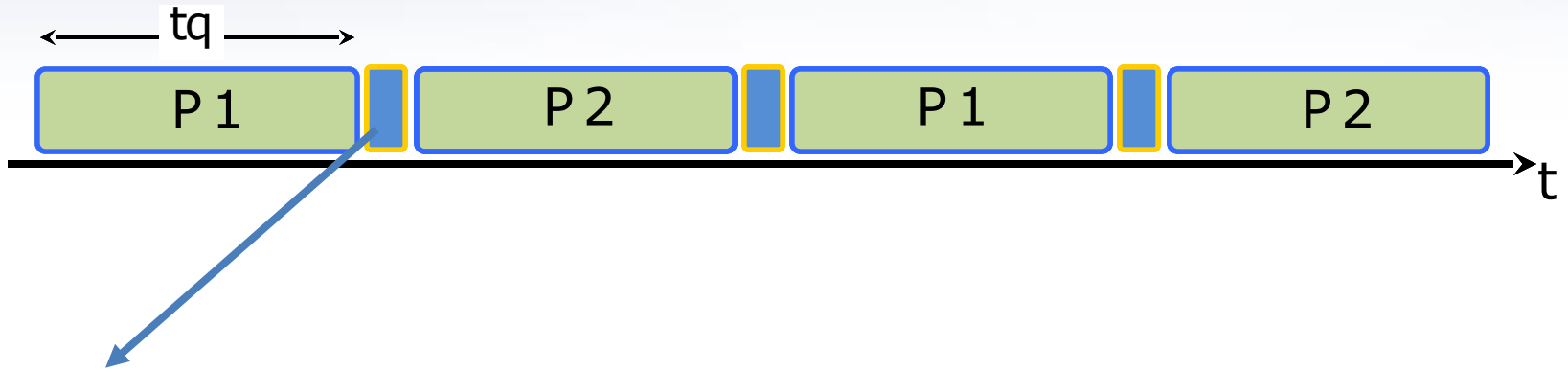
Θεωρία Εργ. Άσκησης 1

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.

```
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;
}
```

Διαχείριση αρχείων

Παράδειγμα 4: Άνοιγμα και εγγραφή σε αρχείο

```
int fd = open(argv[1], O_CREAT | O_APPEND | O_WRONLY, 0644);

if (fd == -1) {
        perror("open");
        return 1;
 }

if (write(fd, buf, strlen(buf)) < strlen(buf)) {
        perror("write");
        return 1;
}
close(fd);
```

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

Υπεύθυνος εργαστηριακών ασκήσεων

Δρ. Χρήστος Παυλάτος

pavlatos@cslab.ece.ntua.gr

Βοηθοί εργαστηρίου

Άγγελος Κολαϊτης, neoaggelos@gmail.com
Βαγγέλης Μακρής, vaggelismacris@gmail.com
Σταμάτης Κατσαούνης, katsaouniss@gmail.com
Στέλλα Γκίρτσου, *sgirtsou@noa.gr*
Μάριος Κόνιαρης, mkoniari@central.ntua.gr