



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ
Ακαδημαϊκό Έτος 2023-2024
1η Εργαστηριακή Άσκηση

Στοιχεία Φοιτητών:

Ονοματεπώνυμο: **ΚΟΥΤΡΟΥΜΠΕΛΑΣ ΒΑΣΙΛΕΙΟΣ**

ΑΜ: **1093397**

email: up1093397@ac.upatras.gr

Ονοματεπώνυμο: **ΜΙΝΩΠΕΤΡΟΣ ΦΙΛΙΠΠΟΣ**

ΑΜ: **1093431**

email: up1093431@ac.upatras.gr

Περιεχόμενα

Εισαγωγή.....	3
Ερώτημα 1: Shell Scripting.....	3
Ερώτημα 2: Διεργασίες.....	4
Ερώτημα 3: Συγχρονισμός Διεργασιών.....	5
Ερώτημα 4: Χρονοπρογραμματισμός Διεργασιών.....	7

Εισαγωγή

Στις παρακάτω ερωτήσεις υλοποιήθηκαν όλα τα τμήματα της άσκησης και όλα λειτουργούν σωστά με βάση τους ελέγχους που κάναμε.

Ερώτημα 1: Shell Scripting

Το πρόγραμμα που υλοποιήσαμε για το Ερώτημα 1 αποτελείται από ένα κεντρικό `while loop` το οποίο τρέχει μέχρι να επιλέξουμε μέσω του `case` που εμπεριέχεται σε αυτό να κάνουμε έξοδο. Μέσα σε αυτό το `loop` περιλαμβάνεται η εκτύπωση του μενού και η λήψη της επιλογής του χρήστη για το `case statement` που ακολουθεί. Το **case** statement έχει 6 επιλογές όπου η κάθε μια εκτελεί και μια λειτουργία από αυτές που δίνονται στην εκφώνηση.

Για την πρώτη λειτουργία, λαμβάνουμε το `path` του αρχείου επιχειρήσεων από τον χρήστη και το τοποθετούμε στην μεταβλητή όπου πρέπει να είναι αποθηκευμένο αυτό. Αν το `path` δεν υπάρχει, εμφανίζεται ενημερωτικό μήνυμα.

Για την δεύτερη λειτουργία, λαμβάνουμε τον κωδικό μιας επιχείρησης από τον χρήστη και μέσω της εντολής **awk** βρίσκουμε την γραμμή η οποία έχει ίδιο κωδικό επιχείρησης με τον δοθέν και τυπώνουμε τα στοιχεία της.

Για την τρίτη λειτουργία, λαμβάνουμε τον κωδικό μιας επιχείρησης από τον χρήστη και το πεδίο το οποίο θέλει να αλλάξει σε αυτή. Έπειτα μετατρέπουμε το πεδίο αυτό στον αριθμό με τον οποίο εμφανίζεται η στήλη του στο `csv` αρχείο, λαμβάνουμε την νέα τιμή για το πεδίο από τον χρήστη και τυπώνουμε την παλιά μέσω της **awk**. Σε αυτό το σημείο, αντιμετωπίσαμε το πρόβλημα τροποποίησης πολλαπλών στοιχείων χωρίς να χρειαστεί να αποθηκεύσουμε πρώτα το αρχείο. Έτσι, δημιουργούμε κατά την τροποποίηση του πρώτου στοιχείου για ένα αρχείο ένα νέο προσωρινό αρχείο το οποίο είναι αντίγραφο του πρώτου και από την πρώτη τροποποίηση και μετά δουλεύουμε σε αυτό. Άρα για να κάνουμε την επιθυμητή αλλαγή κάθε φορά χρησιμοποιούμε το προσωρινό αρχείο και την εντολή **sed** η οποία εντοπίζει την γραμμή που βρίσκεται ο κωδικός της επιχείρησης και αντικαθιστά με την βοήθεια του αριθμού του πεδίου την κατάλληλη στήλη. Το επόμενο πρόβλημα που αντιμετωπίσαμε είναι η αλλαγή του πεδίου `ID` το οποίο αν αλλάξει προκαλεί πρόβλημα στην εκτύπωση της νέας τιμής μέσω της **awk**. Για την αντιμετώπιση αυτού υλοποιήσαμε σε περίπτωση αλλαγής του `ID` να ενημερώνεται και η μεταβλητή που κρατά τον κωδικό της επιχείρησης.

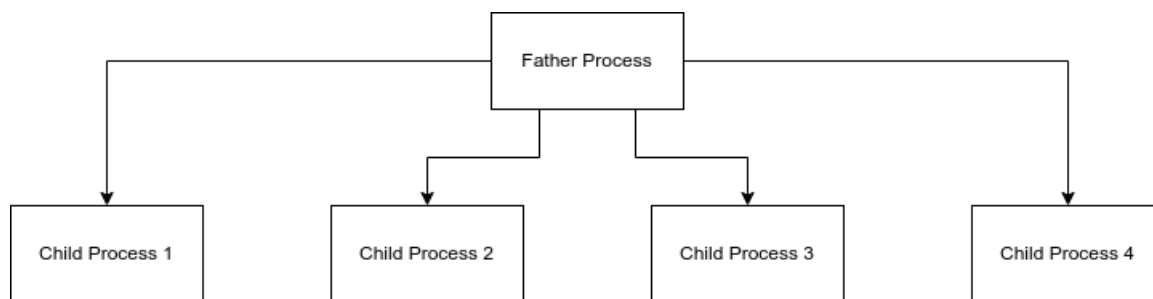
Για την τέταρτη λειτουργία, χρησιμοποιούμε την εντολή **more** για το αρχείο που έχουμε επιλέξει ή έχει υποστεί επεξεργασία.

Για την πέμπτη λειτουργία, λαμβάνουμε το `path` όπου θα αποθηκευτεί το αρχείο από τον χρήστη, αν αυτό δεν δοθεί τότε χρησιμοποιούμε το προεπιλεγμένο. Για την αποθήκευση, αν το αρχείο έχει υποστεί επεξεργασία (δηλ. είναι πλέον το προσωρινό) τότε αποθηκεύεται, αλλιώς δεν γίνεται τίποτα.

Τέλος, η έκτη λειτουργία αφαιρεί το προσωρινό αρχείο και τερματίζει το πρόγραμμα, ενώ αν ο χρήστης δεν έχει δώσει έγκυρη επιλογή στην επιλογή λειτουργίας, του εμφανίζεται ενημερωτικό μήνυμα και μπορεί να επιλέξει ξανά.

Ερώτημα 2: Διεργασίες

Τα προγράμματα που υλοποιήσαμε για το Ερώτημα 2 αποτελούνται σε μεγάλο μέρος τους από το πρόγραμμα της εκφώνησης. Τα κομμάτια που προστέθηκαν ήταν η δημιουργία κοινής μνήμης (shared memory) με την βοήθεια της συνάρτησης **mmap** και η επανάληψη **for** η οποία δημιουργεί τις n διεργασίες-εργάτες μέσω της συνάρτησης **fork**. Στην υλοποίηση μας θέσαμε ως n τον αριθμό 4 μιας και ο υπολογιστής όπου δοκιμάστηκε το πρόγραμμα έχει 4 πυρήνες, έτσι θα δημιουργηθούν 4 διεργασίες-παιδιά. Η κάθε διεργασία μέσα στο **for loop** αν είναι διεργασία παιδί εκτελεί τον ζητούμενο υπολογισμό με **seed** το **PID** της. Τα **PID** των διεργασιών αυτών αποθηκεύονται σε έναν δυναμικό πίνακα ώστε το πρόγραμμα πατέρας να περιμένει να τελειώσουν όλες.



Παράδειγμα δημιουργίας 4 παιδιών από έναν πατέρα

Για την δεύτερη περίπτωση όπου μας ζητείται η προστασία της κοινής μνήμης με σημαφόρους υλοποιούμε το παραπάνω πρόγραμμα με την διαφορά ότι δημιουργούμε έναν σημαφόρο, τον αρχικοποιούμε με την τιμή 1 και τοποθετούμε μια κλήση της **sem_wait** πριν την πρόσβαση μιας διεργασίας-παιδιού στην κρίσιμη περιοχή και μια κλήση της **sem_post** μετά.

Τα αποτελέσματα εκτέλεσης των 3 διαφορετικών αρχείων:

```
> ./integral_mc_seq
Result=0.7386452092778891 Error=2.211241e-06 Rel.Error=2.993653e-06 Time=5.734586 seconds
```

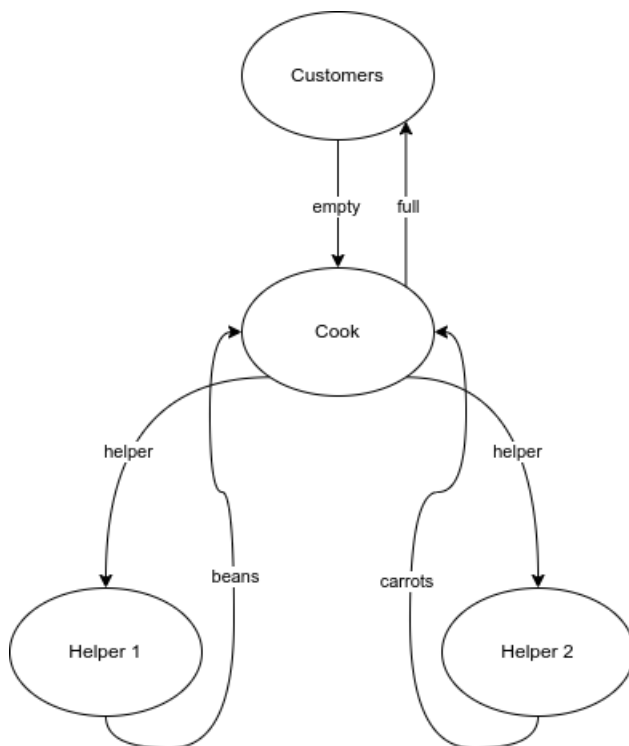
```
> ./integral_mc_shm
Result=0.7321617962038833 Error=6.481202e-03 Rel.Error=8.774471e-03 Time=0.000304 seconds
```

```
> ./integral_mc_shm_sem
Result=0.7357318408714204 Error=2.911157e-03 Rel.Error=3.941224e-03 Time=0.000372 seconds
```

Όπως παρατηρούμε από τα αποτελέσματα, ο χρόνος υπολογισμού για τα 2 τελευταία προγράμματα έχει αυξηθεί κατά πολύ σε σχέση με το δοθέν πρόγραμμα, ωστόσο, το απόλυτο και σχετικό σφάλμα είναι μεγαλύτερο. Επίσης, τα σφάλματα αυτά δεν είναι πάντα σταθερά σε κάθε τρέξιμο, αυτό ίσως οφείλεται στον μικρό αριθμό n και στην χρήση της γεννήτριας ψευδοτυχαίων αριθμών.

Ερώτημα 3: Συγχρονισμός Διεργασιών

Ο παρακάτω ψευδοκώδικας είναι γραμμένος με C-like συντακτικό, ενώ έχουμε κρατήσει τα ονόματα των συναρτήσεων των σημαφόρων ίδια με αυτά της C. Επίσης θεωρούμε ότι η κάθε διεργασία τρέχει παράλληλα και για απεριόριστο χρονικό διάστημα. Για αρχή, αρχικοποιούμε τις global μεταβλητές για τον αριθμό των μερίδων και τα υλικά (ξεχωριστά) και έπειτα δηλώνουμε και αρχικοποιούμε τις τιμές των σημαφόρων μας. Στην υλοποίηση μας έχουμε θεωρήσει πως η κατσαρόλα είναι αρχικά άδεια. Παρακάτω το διάγραμμα που δείχνει το σκεπτικό της ροής των διεργασιών και ο ψευδοκώδικας ($\Sigma X=4$ και $\Sigma Y=3$ για την δική μας περίπτωση).



// variables and semaphores

```
int portions=0;
```

```
int beans=0;
```

```
int carrots=0;
```

```
sem_t customer = sem_open(1);
```

```
sem_t full = sem_open(0);
```

```
sem_t empty = sem_open(0);
```

```
sem_t helper = sem_open(0);
```

```
sem_t beans = sem_open(0);
```

```
sem_t carrots = sem_open(0);
```

```
// customer process
while (1)
{
    sem_post(empty);
    sem_wait(full);
    while (portions>0)
    {
        sem_wait(customer);
        take_food();
        portions--;
        sem_post(customer);
    }
}
```

```
// cook process
while (1)
{
    sem_wait(empty);
    sem_post(helper);
    sem_post(helper);
    sem_wait.beans);
    sem_wait(carrots);
    cook();
    portions=9;
    sem_post(full);
}
```

```
// helper1 process
while (1)
{
    sem_wait(helper);
    while (beans<=6)
    {
        helper();
        beans+=3
    }
    sem_post.beans);
}
```

```
// helper2 process
while (1)
{
    sem_wait(helper);
    while (carrots<=3)
    {
        helper();
    }
}
```

```

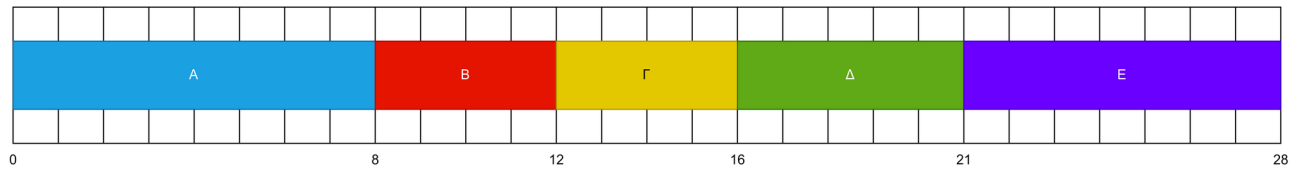
    beans+=3
}
sem_post(carrots);
}

```

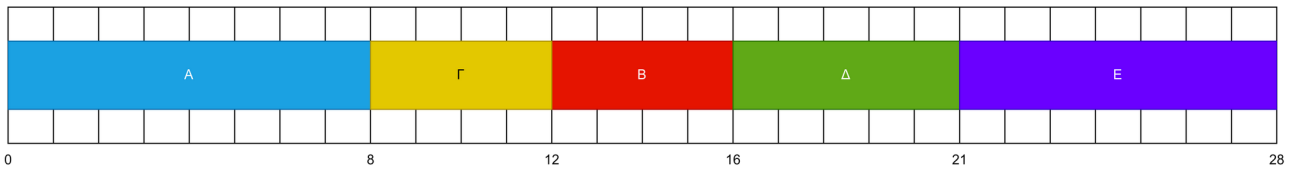
Ερώτημα 4: Χρονοπρογραμματισμός Διεργασιών

(α) Τα διαγράμματα Gantt για τους ζητούμενους αλγορίθμους χρονοδρομολόγησης:

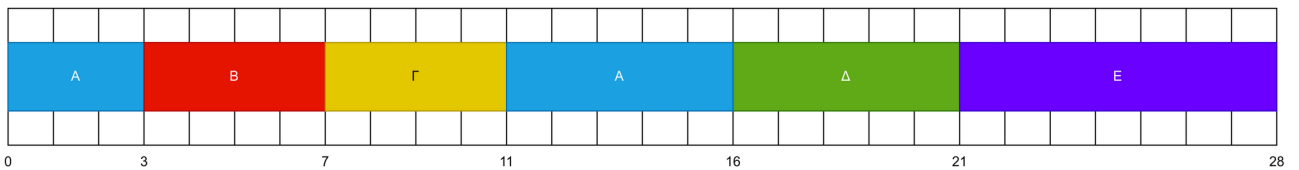
FCFS



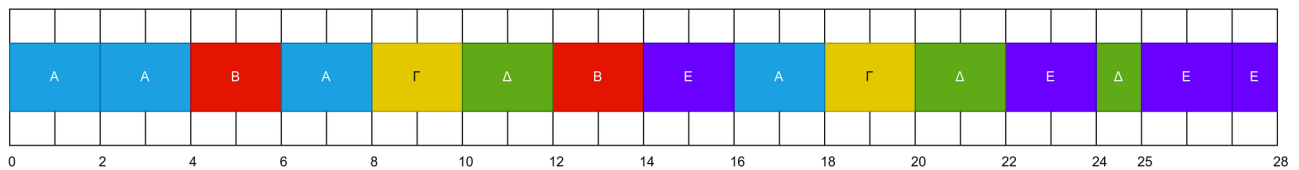
SJF



SRTF



RR



(β) Οι ζητούμενοι υπολογισμοί για τους προηγούμενους αλγόριθμους:

FCFS							
Διεργασία	Χρόνος άφιξης (XA)	Διάρκεια εκτέλεσης (ΔΕ)	PID	Χρόνος αναμονής	Χρόνος απόκρισης	Χρόνος ολοκλήρωσης	Θεματικές εναλλαγές
A	0	8	3	0	0	8	1
B	3	4	2	5	5	9	1
Γ	4	4	1	8	8	12	1
Δ	5	5	4	11	11	16	1
E	7	7	5	14	14	21	1
Μέσος όρος				7.6	7.6	13.2	
Πλήθος							5

SJF							
Διεργασία	Χρόνος άφιξης (XA)	Διάρκεια εκτέλεσης (ΔΕ)	PID	Χρόνος αναμονής	Χρόνος απόκρισης	Χρόνος ολοκλήρωσης	Θεματικές εναλλαγές
A	0	8	3	0	0	8	1
B	3	4	2	9	9	13	1
Γ	4	4	1	4	4	8	1
Δ	5	5	4	11	11	16	1
E	7	7	5	14	14	21	1
Μέσος όρος				7.6	7.6	13.2	
Πλήθος							5

SRTF							
Διεργασία	Χρόνος άφιξης (XA)	Διάρκεια εκτέλεσης (ΔΕ)	PID	Χρόνος αναμονής	Χρόνος απόκρισης	Χρόνος ολοκλήρωσης	Θεματικές εναλλαγές
A	0	8	3	8	0	16	2
B	3	4	2	0	0	4	1
Γ	4	4	1	3	3	7	1
Δ	5	5	4	11	11	16	1
E	7	7	5	14	14	21	1
Μέσος όρος				7.2	5.6	12.8	
Πλήθος							6

RR							
Διεργασία	Χρόνος άφιξης (XA)	Διάρκεια εκτέλεσης (ΔΕ)	PID	Χρόνος αναμονής	Χρόνος απόκρισης	Χρόνος ολοκλήρωσης	Θεματικές εναλλαγές
A	0	8	3	10	0	18	3
B	3	4	2	7	1	11	2
Γ	4	4	1	12	4	16	2
Δ	5	5	4	15	5	20	3
E	7	7	5	14	7	21	3
Μέσος όρος				11.6	3.4	17.2	
Πλήθος							13

(γ) Έχουμε υπολογίσει $MXA_{(SRTF)} = 7.2$ άρα ο $MXA_{(RR \text{ with Context Switch})} = 3 * MXA_{(SRTF)} = 21.6$. Έστω x ο χρόνος εναλλαγής που θέλουμε να βρούμε. Ισχύει ότι, $MXA_{(RR \text{ with Context Switch})} = (XA_A + XA_B + XA_\Gamma + XA_\Delta + XA_E) / 5$ και έχουμε για κάθε διεργασία Y $XA_Y = t_{εξόδου}(Y) - t_{είσοδου}(Y) - t_{CPU}(Y) - \text{Συνολικός_αριθμός_θεματικών_εναλλαγών_από_την_είσοδο}(Y) * x$.

- $XA_A = 18 - 0 - 8 - 7 * x$
- $XA_B = 14 - 3 - 4 - 5 * x$
- $XA_\Gamma = 20 - 4 - 4 - 8 * x$
- $XA_\Delta = 25 - 5 - 5 - 10 * x$
- $XA_E = 28 - 7 - 7 - 10 * x$

$MXA_{(RR \text{ with Context Switch})} = (XA_A + XA_B + XA_\Gamma + XA_\Delta + XA_E) / 5 \Rightarrow 21.6 = (58 - 40 * x) / 5 \Rightarrow x = 50/40$
 $\Rightarrow x = 1.25$ ο χρόνος εναλλαγής.

Το διάγραμμα Gantt:

