

CEID

Λειτουργικά συστήματα

1ο project 2024-2025

Αντωνίου Σωτήριος (1067512)

basotos@hotmail.com

Κωνσταντίνος Χριστάκος (1070903)

up1070903@ac.upatras.gr

1. Shell Scripting

Στην παραπάνω άσκηση προσομοιώσαμε μια κατάσταση όπου σε μία άσκηση εγκατάλειψής πλοίου, με διαχείριση και ανάλυση των δεδομένων των επιβατών που συμμετέχουν στην άσκηση. Φτιάξαμε επιτυχώς ένα αρχείο shell script με όνομα `processes_ipc.sh` σε Bash όπως ζητήθηκε για τα παρακάτω ερωτήματα.

Για να επιλύσουμε το πρώτο ερώτημα φτιάξαμε μία συνάρτηση `insert_data` η οποία έχει την ικανότητα με την εντολή `./processes_ipc.sh insert "/mnt/c/Users/konar/Desktop/Λειτουργικά Συστήματα/PROJECT1/passenger_data1.csv"` να εισάγει κάποιο αρχείο με δεδομένα και αν δεν δοθεί αρχείο η εντολή `./processes_ipc.sh insert` να ζητάει από το πληκτρολόγιο από το πληκτρολόγιο εισαγωγή επιβάτη με τα στοιχεία που μας έχετε δώσει και να τα αποθηκεύει στο αρχείο `passengers.csv` που μας έχει δοθεί.

```
konaras@LAPTOP-H34A5047:/mnt/c/Users/konar/Desktop/Λειτουργικά Συστήματα/PROJECT1$ ./processes_ipc.sh insert
Δεν δόθηκε όνομα αρχείου. Εισάγετε δεδομένα χειροκίνητα.
Εισάγετε τα δεδομένα του επιβάτη (κωδικός; όνομα; ηλικία; χώρα; κατάσταση; διασωθείς): |
```

Για την υλοποίηση του δεύτερου ερωτήματος φτιάξαμε την συνάρτηση `search_passenger` η οποία έχει την ικανότητα με την εντολή `./processes_ipc.sh search "Kostas Christakos"` να ψάχνει μέσα στο αρχείο ή αντί για το ονοματεπώνυμο μπορεί να ψάξει το καθένα ξεχωριστά και μετά να τα εμφανίζει στην οθόνη.

```
konaras@LAPTOP-H34A5047:/mnt/c/Users/konar/Desktop/Λειτουργικά Συστήματα/PROJECT1$ ./processes_ipc.sh search kostas
Στοιχεία επιβάτη:
Κωδικός: 880
Όνομα: Rouvas Kostas
Ηλικία: 53
Χώρα: Bulgaria
Κατηγορία: Crew
Διασωθείς: no
```

Για την υλοποίηση του τρίτου ερωτήματος φτιάξαμε την συνάρτηση `update_passenger` η οποία με την εντολή `./processes_ipc.sh update 123 fullname:"Sotiris Antoniou"` μπορείς να αλλάξεις οποιοδήποτε από τα δοθέν πεδία αλλά ολόκληρο το record.

```
konaras@LAPTOP-H34A5047:/mnt/c/Users/konar/Desktop/Λειτουργικά Συστήματα/PROJECT1$ ./processes_ipc.sh update 700 record:
"700;elisavet Apostolou;40;greece;Passenger;yes;"
Η εγγραφή ενημερώθηκε από:700;elisavet Apostolou;25;greece;Passenger;yes; σε: 700;elisavet Apostolou;40;greece;Passenger;yes;
```

Για την υλοποίηση του τέταρτου ερωτήματος φτιάξαμε την συνάρτηση `display_file` η οποία με την εντολή `./processes_ipc.sh display` εμφανίζει με τη σειρά τους επιβάτες και σωστα ρυθμισμένη έτσι ώστε να εμφανίζει μόνο όσους χωράνε στο παράθυρο του `admin` και για να εκτυπωση παλι πατάμε `space` και άμα θέλουμε να αποχωρήσουμε `q`.

```
konaras@LAPTOP-H34A5047:/mnt/c/Users/konar/Desktop/Λειτουργικά Συστήματα/PROJECT1$ ./processes_ipc.sh display
code;fullname;age;country;status;rescued
1;Nunez Jorge;20;Russia;Passenger;Yes
2;Hartmann Wolfgang;21;Germany;Passenger;Yes
3;Haarhoff Lily Tembo;22;United Kingdom;Passenger;yes
4;Zhang Wei;23;France;Passenger;yes
5;Zhang Yang;24;Italy;Passenger;yes
6;Nguyen Cam;25;Spain;Passenger;yes
7;Znaimer Moses;26;Poland;Passenger;yes
8;Phan Don;27;Ukraine;Passenger;no
9;Takahashi Koji;28;Romania;Passenger;yes
10;Chen Ben;29;Netherlands;Passenger;yes
11;Ngoche Alex Obanda;30;Belgium;Crew;no
12;Kobayashi Ken;31;Czech Republic (Czechia);Passenger;Yes
13;Ben Dhifallah Karim;32;Sweden;Passenger;Yes
14;Santos Carlos;33;Portugal;Passenger;yes
15;Korner Karl;34;Greece;Passenger;yes
16;Fayed Paul;35;Hungary;Passenger;yes
17;Charoenpura Somchai;36;Austria;Passenger;yes
18;Abe Kenji;37;Belarus;Passenger;yes
19;Li Lei;38;Switzerland;Passenger;no
20;Khan Babar;39;Bulgaria;Passenger;yes
21;Williams Jack;40;Serbia;Passenger;yes
22;Chen Hsin;41;Denmark;Crew;no
23;Wagner Richard;42;Finland;Passenger;Yes
24;Basov Irina;43;Norway;Passenger;Yes
25;Bhiari Ammar;44;Slovakia;Passenger;yes
26;Wolf Paul;45;Ireland;Passenger;yes
27;Burns Frank;46;Croatia;Passenger;yes
Πατήστε <space> για συνέχεια ή <q> για έξοδο...|
```

Για την υλοποίηση του πέμπτου ερωτήματος φτιάξαμε την συνάρτηση `generate_report` η οποία διαβάζοντας το αρχείο που του έχουμε δώσει δημιουργεί αναφορες:

- Πρώτη αναφορά είναι το `ages.txt` η οποία ψάχνει όλους του επιβάτες και τους κατατάζει ανα ηλικία και μας βγάζει αυτό το αποτέλεσμα αποθηκευμένο στο αρχείο `ages.txt` που δημιουργεί εκείνη την στιγμή.

```
C: > Users > konar > Desktop > Λειτουργικά Συστήματα > PROJECT1 > ages.txt
1 | 151 0-18
2 | 289 19-35
3 | 260 36-50
4 | 589 51+
5 |
```

Δεύτερη αναφορά είναι percentages.txt στην οποία γίνεται υπολογισμός συμμετεχόντων στη διάσωση για κάθε ηλικιακή ομάδα. Το συγκεκριμένο υλοποιήθηκε με δυο τρόπους για να πάρουμε τα σωστά αποτελέσματα πρώτα έγινε διαχωρισμός στις ηλικιακές ομάδες και μετά φτιάχτηκε ένα temporary αρχείο το οποίο είχαμε φιλτράρει με grep -I για να βρούμε όσους έχουν yes και να κάνουμε και εκεί διαχωρισμό ηλικίας και να κάνουμε τις πράξεις μετα και για τα δυο μαζί για να βγάλουμε το τελικό σωστό αποτέλεσμα.

```
C: > Users > konar > Desktop > Λειτουργικά Συστήματα > PROJECT1 > percentages.txt
```

1	Αναφορά ποσοστών διασωθέντων ανά ηλικιακή ομάδα:			
2				
3	Ηλικιακή Ομάδα	Συνολικοί Επιβάτες	Διασωθέντες	Ποσοστό (%)
4	-----			
5	0-18	144	122	84.72%
6	19-35	289	238	82.35%
7	36-50	260	217	83.46%
8	51+	589	484	82.17%
9	-----			
10	ΣΥΝΟΛΟ	1282	1061	82.76%
11	-----			

Τρίτη αναφορά είναι το avg.txt και υπολογίζει τον μέσω ορο ηλικίας για κάθε κατηγορία επιβατών.

```
C: > Users > konar > Desktop > Λειτουργικά Συστήματα > PROJECT1 > avg.txt
```

1	Crew 46.7241
2	Passenger 47.5373
3	

Τέταρτη αναφορά είναι το rescued.txt το οποίο φιλτράρει το αρχείο με τους επιβάτες και κρατάει μόνο τους διασωθέντες.

```
C: > Users > konar > Desktop > Λειτουργικά Συστήματα > PROJECT1 > rescued.txt
```

1	1;Nunez Jorge;20;Russia;Passenger;Yes
2	2;Hartmann Wolfgang;21;Germany;Passenger;Yes
3	3;Haarhoff Lily Tembo;22;United Kingdom;Passenger;yes
4	4;Zhang Wei;23;France;Passenger;yes
5	5;Zhang Yang;24;Italy;Passenger;yes
6	6;Nguyen Cam;25;Spain;Passenger;yes

2. Συγχρονισμός Διεργασιών και Σημαφόροι

Στην παραπάνω άσκηση προσομοιάζαμε την αποβίβαση και διάσωση των επιβατών ενός πλοίου, μέσω σωστικών λέμβων και θέλαμε να ελέγξουμε τους κανόνες με τους οποίους θα γίνει.

Για αρχή θεωρήσαμε σωστό και εύχρηστο να χρησιμοποιήσουμε threads, με σκοπό:

- κάθε επιβάτης να λειτουργεί αυτόνομα και ταυτόχρονα με τους άλλους, προσπαθώντας να επιβιβαστεί σε μία λέμβο.
- Η χρήση σημαφόρων που μας ενθαρρύνει η εκφώνηση να χρησιμοποιήσουμε χρειάζεται threads.

Καταλήξαμε πως κάθε θρεντ θα προσομοιάζει και έναν επιβάτη.

Έχουμε ήδη λοιπόν την μεταβλήτη (PASSENGERS), που θα μας δείξει τον αριθμό των θρεντς και το πότε θα τελειώσει το πρόγραμμα.

Εν συνέχεια, θεωρήσαμε επόμενο, να προσομοιώσουμε τους σημαφόρους ως βάρκες (BOATS), με μέγιστη χωρητικότητα των αριθμό θέσεων (SEATS). Ένα semaphore δηλαδή θα επιτρέψει σε threads ίσα με τον αριθμό θέσεων να προχωρήσουν ταυτόχρονα στο πρόγραμμα (δηλαδή στην επιβίβαση), ενώ τα άλλα περιμένουν να αδειάσει κάποια θέση. Δημιουργήσαμε έναν πίνακα σημαφόρων ώστε να μεταβάλλεται ο αριθμός τους ανάλογα με τις ανάγκες του χρήστη για βάρκες (σημαφόρους) χωρίς να χρειάζεται να τους δημιουργούμε ή να τους αφαιρούμε χειροκίνητα κάθε φορά που τρέχουμε το πρόγραμμα με διαφορετικό αριθμό.

```
(extern sem_t* seat_sems[]; // Shared array for boat semaphores)
```

Κάθε thread αφού αρχικοποιηθούν όλα προσπαθεί συνεχώς να επιβιβαστεί σε μία βάρκα (αν έχουμε πολλές προσπαθεί να επιβιβαστεί με σειρά σε όλες), περιμένοντας κάποιο χρόνο αν δεν τα καταφέρει.

Οι “επιβάτες μπαίνουν στην βάρκα και ξεκινάει το ταξίδι τους στου οποίου το τέλος αποβιβάζονται απελευθερώνοντας τις θέσεις τους, και οι βάρκες επιστρέφουν για να παραλάβουν καινούργια θρεντς. Αυτή η διαδικασία επαναλαμβάνεται μέχρι όλα τα θρεντς να επιβιβαστούν σε μία βάρκα και να τελειώσουν το “process” τους, όπου και λαμβάνουμε το αουτπντ ότι το πρόγραμμα τελείωσε.

Τελευταίο βήμα ήταν να μοιράσουμε το αρχείο που άρχισε ως 1 σε 3 διαφορετικά.

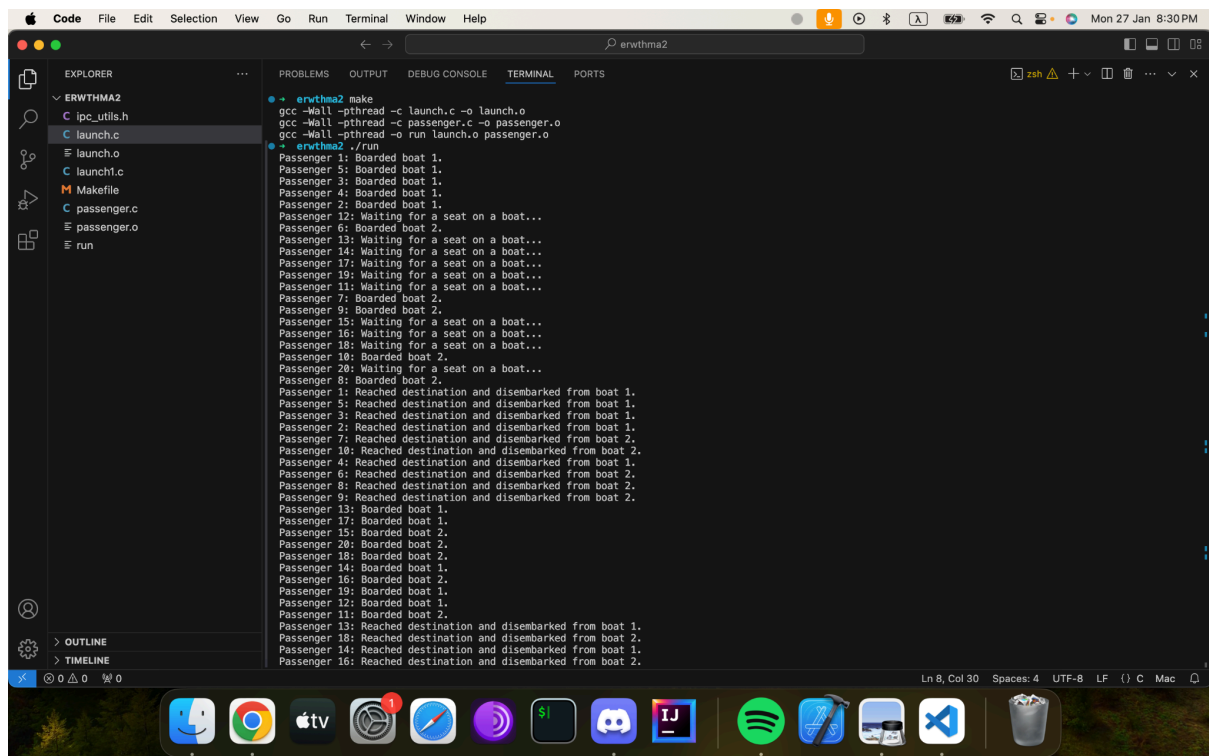
Νομίζω το πρόγραμμα είναι αρκετά απλό και ένα απλό τρέξιμο του προγράμματος

μας δίνει αρκετές πληροφορίες. Έχουμε δημιουργήσει αρχείο Makefile οπότε οι εντολές που θα πατήσουμε είναι:

- 1) make
- 2) ../run

Αν θέλουμε να αλλάξουμε κάποια απ'τις μεταβλητές βρίσκονται στην αρχή του launch.c.

Στις παρακάτω φωτογραφίες έτρεξα το πρόγραμμα για 20 επιβάτες με 2 βάρκες και 5 θέσεις η κάθε μία:



```
erwthma2 make
gcc -Wall -pthread -c launch.c -o launch.o
gcc -Wall -pthread -c passenger.c -o passenger.o
gcc -Wall -pthread -o run launch.o passenger.o

erwthma2 ../run
Passenger 1: Boarded boat 1.
Passenger 5: Boarded boat 1.
Passenger 3: Boarded boat 1.
Passenger 4: Boarded boat 1.
Passenger 2: Boarded boat 1.
Passenger 12: Waiting for a seat on a boat...
Passenger 6: Boarded boat 2.
Passenger 13: Waiting for a seat on a boat...
Passenger 14: Waiting for a seat on a boat...
Passenger 17: Waiting for a seat on a boat...
Passenger 19: Waiting for a seat on a boat...
Passenger 11: Waiting for a seat on a boat...
Passenger 7: Boarded boat 2.
Passenger 9: Boarded boat 2.
Passenger 15: Waiting for a seat on a boat...
Passenger 16: Waiting for a seat on a boat...
Passenger 18: Waiting for a seat on a boat...
Passenger 10: Boarded boat 2.
Passenger 20: Waiting for a seat on a boat...
Passenger 8: Boarded boat 2.
Passenger 1: Reached destination and disembarked from boat 1.
Passenger 5: Reached destination and disembarked from boat 1.
Passenger 3: Reached destination and disembarked from boat 1.
Passenger 2: Reached destination and disembarked from boat 1.
Passenger 7: Reached destination and disembarked from boat 2.
Passenger 10: Reached destination and disembarked from boat 2.
Passenger 4: Reached destination and disembarked from boat 1.
Passenger 6: Reached destination and disembarked from boat 2.
Passenger 8: Reached destination and disembarked from boat 2.
Passenger 9: Reached destination and disembarked from boat 2.
Passenger 13: Boarded boat 1.
Passenger 17: Boarded boat 1.
Passenger 15: Boarded boat 2.
Passenger 20: Boarded boat 2.
Passenger 18: Boarded boat 2.
Passenger 14: Boarded boat 1.
Passenger 16: Boarded boat 2.
Passenger 19: Boarded boat 1.
Passenger 12: Boarded boat 1.
Passenger 11: Boarded boat 2.
Passenger 13: Reached destination and disembarked from boat 1.
Passenger 18: Reached destination and disembarked from boat 2.
Passenger 14: Reached destination and disembarked from boat 1.
Passenger 16: Reached destination and disembarked from boat 2.
Passenger 19: Reached destination and disembarked from boat 1.
Passenger 12: Reached destination and disembarked from boat 1.
Passenger 11: Reached destination and disembarked from boat 2.
Passenger 17: Reached destination and disembarked from boat 1.
Passenger 15: Reached destination and disembarked from boat 2.
Passenger 20: Reached destination and disembarked from boat 2.
All passengers have been saved!
```

```
→ erwthma2
```

προβλήματα:

- 1) αντιμετωπίσαμε πρόβλημα κατά την δημιουργία των σημαφόρων γιατί προσπαθήσαμε να χρησιμοποιήσουμε εντολές (sem_init, sem_destroy) που βρήκαμε στο διαδίκτυο (youtube, google) οι οποίες δεν μας έτρεχαν και δεν μπορούσαμε να καταλάβουμε που κάνουμε λάθος. Εν τέλει χρησιμοποιήσαμε εντολές που βρήκαμε στα έγγραφα του μαθήματος στο e-class.
- 2) Μας παίδεψε λίγο η δημιουργία πολλών σημαφόρων και η σωστή λειτουργία τους, αλλά με ένα for loop καταφέρνουμε να επιβιβάζονται οι επιβάτες σε όλες τις διαθέσιμες βάρκες και με έναν πίνακα καταφέρνουμε να δημιουργήσουμε όλες βάρκες χρειαζόμαστε

3.Χρονοπρογραμματισμός Διεργασιών και Διαχείριση Μνήμης

Η παραπάνω υλοποίηση είναι ένα απλό simulation ενός round robin, με first fit για την κατανομή της μνήμης.

Η μνήμη αναπαριστάται από έναν πίνακα memory με 512 blocks. Με το first fit, κάθε φορά που προσθέτουμε μια διεργασία αναζητούμε αν υπάρχει διαθέσιμο μπλοκ στην μνήμη για να καλυφθούν τα requirements της διεργασίας και αν ναι την προσθέτουμε, δεσμεύουμε το τμήμα και αποθηκεύουμε ότι η διεργασία το χρησιμοποιεί.

Με το RR κάθε διεργασία που βρίσκεται στην μνήμη εκτελείται για ένα κβάντο 3ns, αφαιρούνται απ'τον υπολειπόμενο χρόνο της τα 3ns και προχωράμε στην επόμενη, αφού πρώτα τυπώσουμε τις καταστάσεις cpu, μνήμης και διεργασιών για να δείξουμε πως λειτουργεί.

Ορίστε τι περιέχει:

- Χρησιμοποιήσαμε 2 δομές, μία για διεργασίες και μία για την μνήμη:
 1. typedef struct { int pid; int arrival_time; int duration; int remaining_time; int memory_needed; bool in_memory; } Process;
 2. typedef struct { int start; int size; bool free; int pid; // Process ID occupying this block (-1 if free) } MemoryBlock;
- Αρχικοποιούμε την μνήμη ώστε να είναι ελεύθερη και όλα τα μπλοκς να έχουν μέγεθος 1:
void initialize_memory(MemoryBlock memory[])

- προσπαθούμε να βρούμε μια περιοχή με αρκετό memory για να γίνει μια διεργασία:
int allocate_memory(MemoryBlock memory[], int pid, int size)
- αποδεσμεύουμε το μπλοκ μνήμης όταν τελειώσει η διεργασία:
void deallocate_memory(MemoryBlock memory[], int pid)
- round robin:
void simulate_round_robin(Process processes[], int n, MemoryBlock memory[], int memory_size)
- εκτύπωση κατάστασης μνήμης:
void print_compact_memory_state(MemoryBlock memory[], int memory_size)

Παραθέτω ένα παράδειγμα χρήσης για 4 διεργασίες:

```

erwthma3 ./a.out
Enter the number of processes: 4
Enter details for process 1 (arrival_time duration memory_needed): 0 10 100
Enter details for process 2 (arrival_time duration memory_needed): 6 15 250
Enter details for process 3 (arrival_time duration memory_needed): 2 12 200
Enter details for process 4 (arrival_time duration memory_needed): 3 15 150
Time 0: Process 1 loaded into memory
Time 3:
CPU: Executing Process 1 for 3 ms, remaining time: 7
Memory:
  0-99: Process 1
  100-511: Free
Time 3: Process 3 loaded into memory
Time 6:
CPU: Executing Process 3 for 3 ms, remaining time: 9
Memory:
  0-99: Process 1
  100-299: Process 3
  300-511: Free
Time 6: Process 4 loaded into memory
Time 9:
CPU: Executing Process 4 for 3 ms, remaining time: 12
Memory:
  0-99: Process 1
  100-299: Process 3
  300-449: Process 4
  450-511: Free
Time 12:
CPU: Executing Process 1 for 3 ms, remaining time: 4
Memory:
  0-99: Process 1
  100-299: Process 3
  300-449: Process 4
  450-511: Free
Time 12: Not enough memory for process 2
Time 15:
CPU: Executing Process 3 for 3 ms, remaining time: 6
Memory:
  0-99: Process 1
  100-299: Process 3
  300-449: Process 4
  450-511: Free
Time 18:
CPU: Executing Process 3 for 3 ms, remaining time: 3
Memory:
  0-99: Process 1
  100-299: Process 3
  300-449: Process 4
  450-511: Free
Time 21:
CPU: Executing Process 3 for 3 ms, remaining time: 0
Memory:
  0-99: Process 1
  100-299: Process 3
  300-449: Process 4
  450-511: Free
Time 21: Process 3 finished execution
Time 21: Process 1 finished execution
Time 21: Process 4 finished execution
Time 21: Process 2 not executed due to lack of memory

```



```

Time 18:
CPU: Executing Process 4 for 3 ms, remaining time: 9
Memory:
0-99: Process 1
100-299: Process 3
300-449: Process 4
450-511: Free
Time 21:
CPU: Executing Process 1 for 3 ms, remaining time: 1
Memory:
0-99: Process 1
100-299: Process 3
300-449: Process 4
450-511: Free
Time 21: Not enough memory for process 2
Time 24:
CPU: Executing Process 3 for 3 ms, remaining time: 3
Memory:
0-99: Process 1
100-299: Process 3
300-449: Process 4
450-511: Free
Time 27:
CPU: Executing Process 4 for 3 ms, remaining time: 6
Memory:
0-99: Process 1
100-299: Process 3
300-449: Process 4
450-511: Free
Time 28:
CPU: Executing Process 1 for 1 ms, remaining time: 0
Memory:
0-99: Process 1
100-299: Process 3
300-449: Process 4
450-511: Free
Process 1 completed
Time 28: Not enough memory for process 2
Time 31:
CPU: Executing Process 3 for 3 ms, remaining time: 0
Memory:
0-99: Free
100-299: Process 3
300-449: Process 4
450-511: Free
Process 3 completed
Time 34:
CPU: Executing Process 4 for 3 ms, remaining time: 3
Memory:

Time 34:
CPU: Executing Process 4 for 3 ms, remaining time: 3
Memory:
0-299: Free
300-449: Process 4
450-511: Free
Time 34: Process 2 loaded into memory
Time 37:
CPU: Executing Process 2 for 3 ms, remaining time: 12
Memory:
0-249: Process 2
250-299: Free
300-449: Process 4
450-511: Free
Time 40:
CPU: Executing Process 4 for 3 ms, remaining time: 0
Memory:
0-249: Process 2
250-299: Free
300-449: Process 4
450-511: Free
Process 4 completed
Time 43:
CPU: Executing Process 2 for 3 ms, remaining time: 9
Memory:
0-249: Process 2
250-511: Free
Time 46:
CPU: Executing Process 2 for 3 ms, remaining time: 6
Memory:
0-249: Process 2
250-511: Free
Time 49:
CPU: Executing Process 2 for 3 ms, remaining time: 3
Memory:
0-249: Process 2
250-511: Free
Time 52:
CPU: Executing Process 2 for 3 ms, remaining time: 0
Memory:
0-249: Process 2
250-511: Free
Process 2 completed
→ erwthma3

```

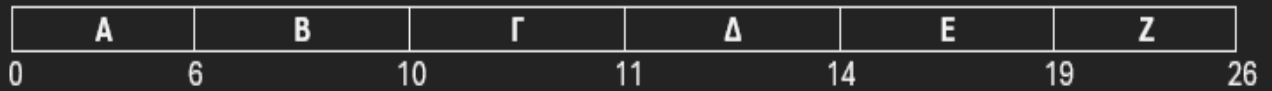
προβλήματα:

Δεν αντιμετωπίσαμε κάποιο αξιοσημείωτο πρόβλημα σ' αυτήν την άσκηση.

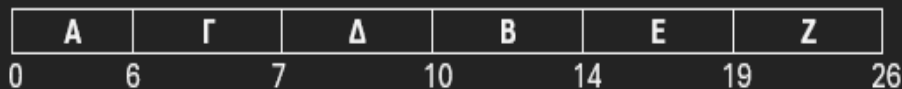
4. Χρονοπρογραμματισμός Διεργασιών

(α) Διάγραμμα Gantt

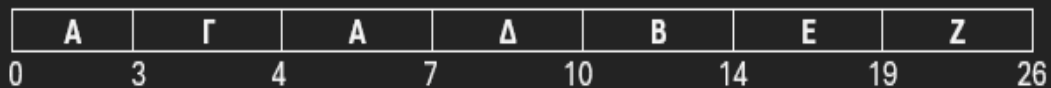
1. FCFS (First Come First Served):



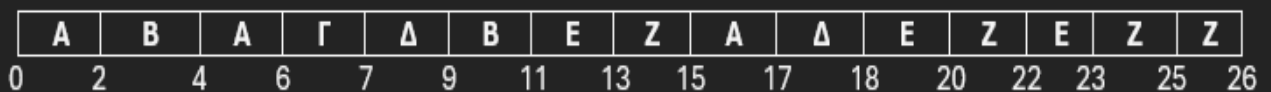
2. SJF (Shortest Job First):



3. SRTF (Shortest Remaining Time First):



4. RR (Round Robin) με κβάντο χρόνου 2:



(β) Υπολογισμοί

1. FCFS (First Come First Served):

Χρόνοι για κάθε διεργασία

- A: Χρόνος Αναμονής = 0, Χρόνος Απόκρισης = 6, Χρόνος Ολοκλήρωσης = 6
- B: Χρόνος Αναμονής = 4, Χρόνος Απόκρισης = 8, Χρόνος Ολοκλήρωσης = 10
- Γ: Χρόνος Αναμονής = 7, Χρόνος Απόκρισης = 8, Χρόνος Ολοκλήρωσης = 11
- Δ: Χρόνος Αναμονής = 7, Χρόνος Απόκρισης = 8, Χρόνος Ολοκλήρωσης = 14
- Ε: Χρόνος Αναμονής = 9, Χρόνος Απόκρισης = 14, Χρόνος Ολοκλήρωσης = 19

- Z: Χρόνος Αναμονής = 13, Χρόνος Απόκρισης = 20, Χρόνος Ολοκλήρωσης = 26

Μέσοι Χρόνοι

- Μέσος Χρόνος Αναμονής = $(0 + 4 + 7 + 7 + 9 + 13) / 6 = 6.67 \text{ ms}$
- Μέσος Χρόνος Απόκρισης = $(6 + 8 + 8 + 8 + 14 + 20) / 6 = 10.66 \text{ ms}$
- Μέσος Χρόνος Ολοκλήρωσης = $(6 + 10 + 11 + 14 + 19 + 26) / 6 = 14.33 \text{ ms}$
- Πλήθος Θεματικών Εναλλαγών: 5

2. SJF (Shortest Job First):

Χρόνοι για κάθε διεργασία

- A: Χρόνος Αναμονής = 0, Χρόνος Απόκρισης = 6, Χρόνος Ολοκλήρωσης = 6
- B: Χρόνος Αναμονής = 8, Χρόνος Απόκρισης = 12, Χρόνος Ολοκλήρωσης = 14
- Γ: Χρόνος Αναμονής = 3, Χρόνος Απόκρισης = 4, Χρόνος Ολοκλήρωσης = 7
- Δ: Χρόνος Αναμονής = 3, Χρόνος Απόκρισης = 6, Χρόνος Ολοκλήρωσης = 10
- E: Χρόνος Αναμονής = 9, Χρόνος Απόκρισης = 14, Χρόνος Ολοκλήρωσης = 19
- Z: Χρόνος Αναμονής = 13, Χρόνος Απόκρισης = 20, Χρόνος Ολοκλήρωσης = 26

Μέσοι Χρόνοι

- Μέσος Χρόνος Αναμονής = $(0 + 8 + 3 + 3 + 9 + 13) / 6 = 6 \text{ ms}$
- Μέσος Χρόνος Απόκρισης = $(6 + 12 + 4 + 6 + 14 + 20) / 6 = 10.33 \text{ ms}$
- Μέσος Χρόνος Ολοκλήρωσης = $(6 + 14 + 7 + 10 + 19 + 26) / 6 = 13.66 \text{ ms}$
- Πλήθος Θεματικών Εναλλαγών: 5

3. SRTF (Shortest Remaining Time First):

Χρόνοι για κάθε διεργασία

- A: Χρόνος Αναμονής = 1, Χρόνος Απόκρισης = 7, Χρόνος Ολοκλήρωσης = 7
- B: Χρόνος Αναμονής = 8, Χρόνος Απόκρισης = 12, Χρόνος Ολοκλήρωσης = 14

- Γ: Χρόνος Αναμονής = 0, Χρόνος Απόκρισης = 1, Χρόνος Ολοκλήρωσης = 4
- Δ: Χρόνος Αναμονής = 3, Χρόνος Απόκρισης = 6, Χρόνος Ολοκλήρωσης = 10
- Ε: Χρόνος Αναμονής = 9, Χρόνος Απόκρισης = 14, Χρόνος Ολοκλήρωσης = 19
- Ζ: Χρόνος Αναμονής = 13, Χρόνος Απόκρισης = 20, Χρόνος Ολοκλήρωσης = 26

Μέσοι Χρόνοι

- Μέσος Χρόνος Αναμονής = $(1 + 8 + 0 + 3 + 9 + 13) / 6 = 5.66 \text{ ms}$
- Μέσος Χρόνος Απόκρισης = $(7 + 12 + 1 + 6 + 14 + 20) / 6 = 10 \text{ ms}$
- Μέσος Χρόνος Ολοκλήρωσης = $(7 + 14 + 4 + 10 + 19 + 26) / 6 = 13.33 \text{ ms}$
- Πλήθος Θεματικών Εναλλαγών: 6

4. RR (Round Robin) με κβάντο χρόνου 2:

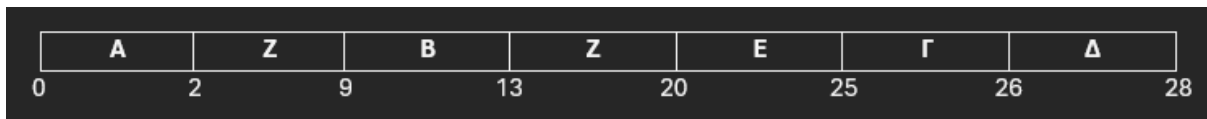
Χρόνοι για κάθε διεργασία

- Α: Χρόνος Αναμονής = 11, Χρόνος Απόκρισης = 17, Χρόνος Ολοκλήρωσης = 17
- Β: Χρόνος Αναμονής = 5, Χρόνος Απόκρισης = 9, Χρόνος Ολοκλήρωσης = 11
- Γ: Χρόνος Αναμονής = 3, Χρόνος Απόκρισης = 4, Χρόνος Ολοκλήρωσης = 7
- Δ: Χρόνος Αναμονής = 11, Χρόνος Απόκρισης = 14, Χρόνος Ολοκλήρωσης = 18
- Ε: Χρόνος Αναμονής = 13, Χρόνος Απόκρισης = 18, Χρόνος Ολοκλήρωσης = 23
- Ζ: Χρόνος Αναμονής = 13, Χρόνος Απόκρισης = 20, Χρόνος Ολοκλήρωσης = 26

Μέσοι Χρόνοι

- Μέσος Χρόνος Αναμονής = $(11 + 5 + 3 + 11 + 13 + 13) / 6 = 9.33 \text{ ms}$
- Μέσος Χρόνος Απόκρισης = $(17 + 9 + 4 + 14 + 18 + 20) / 6 = 13.66 \text{ ms}$
- Μέσος Χρόνος Ολοκλήρωσης = $(17 + 11 + 7 + 18 + 23 + 26) / 6 = 17 \text{ ms}$
- Πλήθος Θεματικών Εναλλαγών: 14

(γ) LRTFP (Longest Remaining Time First Preemptive):



Χρόνοι για κάθε διεργασία

- A: Χρόνος Αναμονής = 0, Χρόνος Απόκρισης = 0, Χρόνος Ολοκλήρωσης = 2
- B: Χρόνος Αναμονής = 7, Χρόνος Απόκρισης = 7, Χρόνος Ολοκλήρωσης = 13
- Γ: Χρόνος Αναμονής = 22, Χρόνος Απόκρισης = 22, Χρόνος Ολοκλήρωσης = 26
- Δ: Χρόνος Αναμονής = 21, Χρόνος Απόκρισης = 21, Χρόνος Ολοκλήρωσης = 28
- E: Χρόνος Αναμονής = 15, Χρόνος Απόκρισης = 15, Χρόνος Ολοκλήρωσης = 25
- Z: Χρόνος Αναμονής = 7, Χρόνος Απόκρισης = 7, Χρόνος Ολοκλήρωσης = 20

Μέσοι Χρόνοι

- Μέσος Χρόνος Αναμονής = $(0 + 7 + 22 + 21 + 15 + 7) / 6 = 12 \text{ ms}$
- Μέσος Χρόνος Απόκρισης = $(0 + 7 + 22 + 21 + 15 + 7) / 6 = 12 \text{ ms}$
- Μέσος Χρόνος Ολοκλήρωσης = $(2 + 13 + 26 + 28 + 25 + 20) / 6 = 19 \text{ ms}$
- Πλήθος Θεματικών Εναλλαγών: 6