2024-2025

Η ομάδα αποτελέιται από τους εξής φοιτητές:

Απόστολος Ζεκυριάς

Σπυρίδων Μανταδάκης

Παναγιώτης Παπανικολάου

Αλέξανδρος Γεώργιος Χαλαμπάκης

**Αναλυτικότερες Πληροφορίες:**

Αλέξανδρος Γεώργιος Χαλαμπάκης 1100754

[up1100754@ac.upatras.gr](mailto:up1100754@ac.upatras.gr)

Φοιτητής 3ου έτους

Σπυρίδων Μανταδάκης 1100613

[up1100613@ac.upatras.gr](mailto:up1100613@ac.upatras.gr)

Φοιτητής 3ου έτους

Απόστολος Ζεκυριάς 1100554

[up1100554@ac.upatras.gr](mailto:up1100554@ac.upatras.gr)

Φοιτητής 3ου έτους

Παναγιώτης Παπανικολάου 1104804

[up1104804@ac.upatras.gr](mailto:up1104804@ac.upatras.gr)

Φοιτητής 3ου έτους

Απαντήσεις στην 1η Εργαστηριακή Άσκηση (Project)

Λειτουργικά Συστήματα

2024 – 2025

**Γενικές Πληροφορίες**

Σχόλια: Κατά την ανάπτυξη των προγραμμάτων μας, όλα τα ερωτήματα υλοποιήθηκαν επιτυχώς και οι κώδικες λειτούργησαν ορθά. Κατά τη διαδικασία, δεν προέκυψαν σοβαρά προβλήματα. Ωστόσο, αντιμετωπίσαμε δυσκολίες στο Ερώτημα 1, καθώς χρειάστηκε να τροποποιήσουμε τον κώδικα που είχαμε δημιουργήσει αφού βασιζόταν στη χρήση μενού. Επιπλέον, στο Ερώτημα 3, και συγκεκριμένα στον διαμοιρασμό της ελεύθερης μνήμης σε blocks, η υλοποίηση της συγκεκριμένης λειτουργικότητας μας απασχόλησε για μεγάλο χρονικό διάστημα, λόγω της χρήσης λανθασμένης προσέγγισης κατά την εκχώρηση των δεδομένων.

Στις επόμενες σελίδες παρουσιάζονται οι απαντήσεις της ομάδας μας στο 1ο Project του μαθήματος.

Περιεχόμενα

[1. Shell Scripting 4](#_Toc188728787)

[**Εισαγωγή δεδομένων στην εφαρμογή** 4](#_Toc188728788)

[Screenshot Εκτέλεσης 4](#_Toc188728789)

[**Προβολή Στοιχείων Επιβαίνοντα** 5](#_Toc188728790)

[Screenshot Εκτέλεσης 5](#_Toc188728791)

[**Αλλαγή Στοιχείων Επιβαίνοντα** 5](#_Toc188728792)

[Screenshot Εκτέλεσης 6](#_Toc188728793)

[**Προβολή Αρχείου** 6](#_Toc188728794)

[Screenshot Εκτέλεσης 7](#_Toc188728795)

[**Δημιουργία αναφορών** 8](#_Toc188728796)

[Screenshot Εκτέλεσης 8](#_Toc188728797)

[2. Συγχρονισμός Διεργασιών και Σημαφόροι 11](#_Toc188728798)

[Βήματα Υλοποίησης: 11](#_Toc188728799)

[**Προβλήματα κατά την υλοποίηση** 11](#_Toc188728800)

[Screenshot Εκτέλεσης 12](#_Toc188728801)

[3. Χρονοπρογραμματισμός Διεργασιών και Διαχείριση Μνήμης 13](#_Toc188728802)

[1. Δομές Δεδομένων 13](#_Toc188728803)

[ Process: 13](#_Toc188728804)

[ MemoryBlock 13](#_Toc188728805)

[**2.** Αρχικοποίηση 13](#_Toc188728806)

[3. Διαχείριση Μνήμης 13](#_Toc188728807)

[ Κατανομή μνήμης: 13](#_Toc188728808)

[ Αποδέσμευση μνήμης: 13](#_Toc188728809)

[4. Προσομοίωση Round Robin 13](#_Toc188728810)

[5**.** Εκτύπωση Καταστάσεων 14](#_Toc188728811)

[6. Εκτέλεση Προγράμματος 14](#_Toc188728812)

[ΠΑΡΑΔΕΙΓΜΑΤΑ ΛΕΙΤΟΥΡΓΙΑΣ 14](#_Toc188728813)

[Process 1 ( 14](#_Toc188728814)

[Process 2 14](#_Toc188728815)

[Παράδειγμα 2. 15](#_Toc188728816)

[Process 1: 16](#_Toc188728817)

[Process 2: 16](#_Toc188728818)

[Process 3: 16](#_Toc188728819)

[Process 4: 16](#_Toc188728820)

[Process 5: 16](#_Toc188728821)

[4. Χρονοπρογραμματισμός Διεργασιών 18](#_Toc188728822)

# Shell Scripting

## **Εισαγωγή δεδομένων στην εφαρμογή**

Συμβουλευτήκαμε διάφορες έγκυρες πηγές από το διαδίκτυο προκειμένου να εξοικειωθούμε με τη χρήση εντολών Linux, όπως οι sed, awk, grep, less, και άλλες. Η εξοικείωση με αυτές τις εντολές κρίθηκε απαραίτητη, καθώς αποτελούν βασικά εργαλεία για την υλοποίηση των ζητούμενων στα πλαίσια του ερωτήματος.

Το πρόγραμμα αναπτύχθηκε ώστε να διαχειρίζεται δεδομένα επιβατών από δύο πηγές:

**Αρχείο**: Αν δοθεί ένα αρχείο passenger\_data.csv, τα δεδομένα του εισάγονται στην εφαρμογή.

**Πληκτρολόγιο**: Αν δε δοθεί αρχείο, το πρόγραμμα επιτρέπει την εισαγωγή δεδομένων μέσω του πληκτρολογίου.

Αν υπάρχει το αρχείο που δόθηκε ως είσοδος, αντιγράφεται το περιεχόμενό του στο αρχείο passengers.csv, ενώ αν δεν υπάρχει, ζητείται η εισαγωγή δεδομένων από τον χρήστη στη μορφή:

[code];[fullname];[age];[country];[status (Passenger/Crew)];[rescued (Yes/No)]

Τα δεδομένα που εισάγει ο χρήστης αποθηκεύονται γραμμή-γραμμή στο αρχείο passengers.csv και η εισαγωγή δεδομένων από το πληκτρολόγιο ολοκληρώνεται μόλις ο χρήστης πληκτρολογήσει τη λέξη exit.

**Έλεγχος ύπαρξης αρχείου:** Χρησιμοποιήθηκε η εντολή if [ -f "$input\_file" ] για να ελέγξουμε αν το αρχείο υπάρχει πριν την αντιγραφή του περιεχομένου.

**Εισαγωγή μέσω πληκτρολογίου:** Αντιμετωπίστηκε η πρόκληση να διακόπτεται η εισαγωγή δεδομένων μέσω της λέξης exit. Αυτό υλοποιήθηκε με βρόχο while και έλεγχο στη λέξη exit.

### Screenshot Εκτέλεσης

A black background with white text

Description automatically generated

## **Προβολή Στοιχείων Επιβαίνοντα**

Το πρόγραμμα υλοποιεί τη δυνατότητα αναζήτησης και προβολής των στοιχείων ενός επιβαίνοντα με βάση το όνομα ή το επώνυμό του. Τα δεδομένα λαμβάνονται από το αρχείο passengers.csv. Για την την υλοποίηση περιλαμβάνει ακολουθούνται τα εξής βήματα:

1. **Ανάγνωση αρχείου**: Το αρχείο passengers.csv φορτώνεται στη μνήμη.
2. **Αναζήτηση χρήστη**: Ζητείται από τον χρήστη να εισάγει το όνομα ή το επώνυμο του επιβαίνοντα.
3. **Φιλτράρισμα αποτελεσμάτων**: Ο αλγόριθμος αναζητά γραμμές του αρχείου που περιέχουν το όνομα ή το επώνυμο, ανεξάρτητα από πεζά/κεφαλαία γράμματα.
4. **Προβολή αποτελεσμάτων**: Εμφανίζονται στην οθόνη τα στοιχεία του επιβαίνοντα, αν βρεθεί, διαφορετικά εμφανίζεται μήνυμα ότι δεν εντοπίστηκε.

Για να μην επηρεάζεται η αναζήτηση από τη χρήση πεζών ή κεφαλαίων γραμμάτων, μετατρέπονται όλα τα γράμματα σε πεζά πριν τη σύγκριση.

Για περιπτώσεις όπου υπάρχουν πολλοί επιβάτες με το ίδιο όνομα/επώνυμο το πρόγραμμα εμφανίζει όλες τις σχετικές εγγραφές.

### Screenshot Εκτέλεσης

A black screen with white text

Description automatically generated

## **Αλλαγή Στοιχείων Επιβαίνοντα**

Η λειτουργία επιτρέπει τη διόρθωση συγκεκριμένων πεδίων ή ολόκληρης της εγγραφής ενός επιβαίνοντα με βάση τον κωδικό, το όνομα ή το επώνυμό του. Το πρόγραμμα επεξεργάζεται το αρχείο passengers.csv και ανανεώνει τα δεδομένα όπως απαιτείται.

Η διαδικασία περιλαμβάνει:

1. **Αναζήτηση εγγραφής**: Το πρόγραμμα αναζητά τον επιβάτη βάσει του κριτηρίου που εισάγει ο χρήστης (κωδικός, όνομα ή επώνυμο).
2. **Επιλογή πεδίου ή ολόκληρης εγγραφής για αλλαγή**: Ο χρήστης καθορίζει μέσω ορίσματος το πεδίο προς αλλαγή ή επιλέγει να αντικαταστήσει ολόκληρη την εγγραφή.
3. **Ενημέρωση δεδομένων**: Η συγκεκριμένη αλλαγή πραγματοποιείται και τα παλιά/νέα δεδομένα προβάλλονται στην οθόνη.
4. **Αποθήκευση στο αρχείο**: Το αρχείο ενημερώνεται με τις αλλαγές

**Διαχείριση Εισόδων:** Έγινε έλεγχος ώστε να ανιχνεύεται η λαθος μορφή των δεδομένων που εισάγονται. Σε τέτοιες περιπτώσεις εμφανίζεται μήνυμα σφάλματος.

**Διατήρηση Μορφής Αρχείου:** Το αρχείο ανανεώνεται μόνο αφού επαληθευθεί ότι η νέα εγγραφή είναι σωστή, αποφεύγοντας αλλαγές στη δομή του.

### Screenshot Εκτέλεσης

A computer screen shot of a black screen

Description automatically generated

## **Προβολή Αρχείου**

Σε αυτό το μέρος της άσκησης δίνουμε στον χρήστη την ικανότητα προβολής του αρχείου passengers.csv και την παρουσίαση όλων των εγγραφών του στην οθόνη. Η διαδικασία προβλέπει τη διαχείριση των δεδομένων, ώστε να γεμίζει η οθόνη με συγκεκριμένο αριθμό εγγραφών κάθε φορά. Ο χρήστης μπορεί να συνεχίσει την προβολή πατώντας το πλήκτρο <space>.

##### Βήματα Υλοποίησης:

1. **Άνοιγμα Αρχείου**: Το αρχείο passengers.csv διαβάζεται γραμμή-γραμμή.
2. **Εμφάνιση Εγγραφών**: Οι εγγραφές εμφανίζονται στην οθόνη, μέχρι να γεμίσει η διαθέσιμη χωρητικότητα.
3. **Εισαγωγή Από Χρήστη**: Το πρόγραμμα αναμένει από τον χρήστη να πατήσει <space> για να συνεχίσει ή οποιοδήποτε άλλο πλήκτρο για να διακόψει την προβολή.
4. **Ολοκλήρωση Προβολής**: Όταν όλες οι εγγραφές έχουν εμφανιστεί, το πρόγραμμα ενημερώνει τον χρήστη ότι η προβολή ολοκληρώθηκε.

**Έλεγχος Εισαγωγής Χρήστη:** Ενσωματώθηκε κατάλληλος έλεγχος για την ανίχνευση του πλήκτρου <space> και τη διαχείριση άλλων εισαγωγών.

**Διαχείριση Μεγάλων Αρχείων:** Ορίστηκε κατάλληλο μέγεθος σελίδας (π.χ., 10 γραμμές) για να εξασφαλιστεί η ομαλή εμφάνιση των δεδομένων.

### Screenshot Εκτέλεσης



A screenshot of a computer screen

Description automatically generated

## **Δημιουργία αναφορών**

Σε αυτό το σημείο το πρόγραμμα μας πρέπει να μπορεί να δημιουργίσει αναφορές βασισμένες σε ξεχωριστά χαρακτηριστικά. Επομένως αναλύει τα δεδομένα του αρχείου passengers.csv και παράγει τέσσερις ξεχωριστές αναφορές. Η επεξεργασία πραγματοποιείται σε επιμέρους βήματα, καθένα από τα οποία εξάγει δεδομένα σε ξεχωριστό αρχείο.

**Οι αναφορές που δημιουργούνται είναι:**

1. **Ηλικιακές Ομάδες** (ages.txt):
   * Ταξινόμηση των επιβατών σε ηλικιακές ομάδες: 0-18, 19-35, 36-50, 51+.
   * Καταμέτρηση επιβατών ανά ομάδα.
2. **Ποσοστά Διάσωσης** (percentages.txt):
   * Υπολογισμός του ποσοστού των διασωθέντων για κάθε ηλικιακή ομάδα.
3. **Μέση Ηλικία** (avg.txt):
   * Υπολογισμός της μέσης ηλικίας επιβατών ανά κατηγορία (Passenger ή Crew).
4. **Διασωθέντες** (rescued.txt):
   * Δημιουργία νέου αρχείου με δεδομένα μόνο για τους διασωθέντες.

### Screenshot Εκτέλεσης

A computer screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

# Συγχρονισμός Διεργασιών και Σημαφόροι

Η άσκηση αφορά την προσομοίωση μιας διαδικασίας εγκατάλειψης πλοίου, όπου οι επιβαίνοντες προσπαθούν να επιβιβαστούν σε λέμβους διάσωσης υπό περιορισμούς χωρητικότητας. Η προσέγγιση μας βασίζεται στη χρήση διεργασιών (processes) και σημαφόρων για τον συγχρονισμό των επιβαινόντων και των διαθέσιμων θέσεων.

Κάθε λέμβος έχει περιορισμένο αριθμό θέσεων, ο οποίος ελέγχεται μέσω των σημαφόρων. Οι επιβάτες περιμένουν μέχρι να απελευθερωθεί θέση και στη συνέχεια επιβιβάζονται. Ο συγχρονισμός αυτός εξασφαλίζει ότι δεν θα επιβιβαστούν περισσότεροι επιβάτες από τις διαθέσιμες θέσεις των λέμβων.

## Βήματα Υλοποίησης:

1. Ο κύριος κώδικας δημιουργεί τον σημαφόρο και εκκινεί τις διεργασίες των επιβατών.
2. Κάθε διεργασία ελέγχει τη διαθεσιμότητα θέσης μέσω του σημαφόρου.
3. Οι επιβάτες επιβιβάζονται αν υπάρχει θέση. Αν όχι, περιμένουν.
4. Όταν ένας επιβάτης αποβιβάζεται, απελευθερώνει τη θέση.

Πρέπει να σημειωθεί ότι η εκτέλεση του προγράμματος πραγματοποιείται μέσω του αρχείου launch.c, ενώ τα αρχεία ipc\_utils.h και passenger.c λειτουργούν ως βοηθητικά αρχεία. Ωστόσο, για την ορθή λειτουργία του launch.c, είναι απαραίτητο να έχει προηγηθεί η δημιουργία του εκτελέσιμου αρχείου που αντιστοιχεί στο passenger.c.

## **Προβλήματα κατά την υλοποίηση**

**1ο Πρόβλημα:** Κατά την ανάπτυξη της εφαρμογής αυτής, αντιμετωπίσαμε ένα αρκετα μεγάλο πρόβλημα στη διαχείριση των σημαφόρων στο Windows API. Συγκεκριμένα, διαπιστώσαμε ότι το header <semaphores.h> δεν είναι διαθέσιμο σε συστήματα Windows, καθώς αποτελεί αποκλειστικό στοιχείο των Unix. Αυτό είχε ως αποτέλεσμα την αδυναμία χρήσης threads ή ακόμη και απλών διεργασιών.

Παράλληλα, η δεύτερη προσέγγιση μας που βασίστηκε στη χρήση του header <windows.h>, όπως προτάθηκε από πολλές διαδικτυακές πηγές, δεν απέδωσε τα επιθυμητά αποτελέσματα. Τελικά, καταλήξαμε στη χρήση του header <winbase.h>, το οποίο μας επέτρεψε να υλοποιήσουμε τις απαραίτητες διεργασίες. Ωστόσο, αυτή η επιλογή καθιστά το πρόγραμμα μη εκτελέσιμο σε Unix συστήματα, καθώς βασίζεται εξ ολοκλήρου στο Windows API.

Επιπλέον, παρατηρήθηκε ότι ενώ το πρόγραμμα εκτελέστηκε επιτυχώς στους τρεις από τους τέσσερις υπολογιστές της ομάδας, σε έναν από αυτούς παρουσίασε προβλήματα, η αιτία των οποίων παραμένει αδιευκρίνιστη.

**2ο Πρόβλημα:** Κατά τη διάρκεια της εκτέλεσης, υπήρξαν περιπτώσεις όπου ένας από τους σημαφόρος δεν μπορούσε να ανοίξει. Για να επιλύσουμε το πρόβλημα αυτό προσθέσαμε μηνύματα σφάλματος (π.χ. Error: ΑΔΥΝΑΜΙΑ ΔΗΜΙΟΥΡΓΙΑΣ ΣΗΜΑΦΟΡΟΥ. Error code: %lu\n") και σιγουρευτήκαμε ότι όλοι οι σημαφόροι κλείνουν σωστά στο τέλος.

### Screenshot Εκτέλεσης

A screen shot of a computer

Description automatically generated

# 3. Χρονοπρογραμματισμός Διεργασιών και Διαχείριση Μνήμης

Το πρόγραμμα που υλοποιήσαμε προσομοιώνει **προγραμματισμό διαδικασιών (process scheduling)** με χρήση της **πολιτικής Round Robin (RR)** και διαχείριση μνήμης με στατική κατανομή. Παρακάτω παρατίθεται μια συνοπτική περιγραφή του:

## 1. Δομές Δεδομένων

* Process: Αναπαριστά μια διεργασία και περιλαμβάνει:
  + ID της διεργασίας (pid), χρόνο άφιξης, διάρκεια, εναπομένουσα διάρκεια, απαιτήσεις μνήμης και αν είναι φορτωμένη στη μνήμη.
* MemoryBlock**:** Αναπαριστά ένα τμήμα μνήμης και περιλαμβάνει:
  + Διεύθυνση εκκίνησης, μέγεθος, αν είναι ελεύθερο και το ID της διεργασίας που το κατέχει.

**2.** Αρχικοποίηση

* Η μνήμη αρχικοποιείται ως ένα μεγάλο ελεύθερο μπλοκ 512 KB.
* Οι διεργασίες εισάγονται από τον χρήστη, παρέχοντας:
  + Χρόνο άφιξης, συνολική διάρκεια εκτέλεσης και απαιτούμενη μνήμη.

## 3. Διαχείριση Μνήμης

* Κατανομή μνήμης: Ελέγχεται αν υπάρχει ελεύθερο μπλοκ με αρκετή μνήμη. Αν ναι, καταλαμβάνεται, και αν περισσεύει μνήμη, το μπλοκ "σπάει".
* Αποδέσμευση μνήμης: Απελευθερώνεται η μνήμη όταν ολοκληρωθεί μια διεργασία.

## 4. Προσομοίωση Round Robin

* Προσομοιώνει την εκτέλεση των διεργασιών σε κβάντα χρόνου (**time slices**) των 3ms:
  + Αν μια διεργασία δεν έχει φορτωθεί στη μνήμη, επιχειρείται κατανομή μνήμης.
  + Η διεργασία εκτελείται για το διαθέσιμο **time slice** (ή λιγότερο αν τελειώνει νωρίτερα).
  + Μετά από κάθε **time slice**, ενημερώνονται οι χρόνοι και η μνήμη.
* Αν όλες οι διεργασίες ολοκληρωθούν, η προσομοίωση σταματά.

5**.** Εκτύπωση Καταστάσεων

* Κατά την εκτέλεση:
  + Εμφανίζονται μηνύματα όταν μια διεργασία φορτώνεται/περιμένει μνήμη.
  + ΕΜΦΑΝΙΖΟΝΤΑΙ ΤΑ ΤΜΗΜΑΤΑ ΜΝΗΜΗΣ ΜΕ ΤΗΝ ΚΑΤΑΣΤΑΣΗ ΚΑΘΕ ΜΠΛΟΚ (ΕΛΕΥΘΕΡΟ Η κατειλημμένο).

## 6. Εκτέλεση Προγράμματος

* Ο χρήστης εισάγει:
  + Αριθμό διεργασιών (μέγιστο 5).
  + Τα χαρακτηριστικά κάθε διεργασίας.
* Ξεκινά η προσομοίωση με διαχείριση μνήμης και εκτέλεση διεργασιών.

## ΠΑΡΑΔΕΙΓΜΑΤΑ ΛΕΙΤΟΥΡΓΙΑΣ

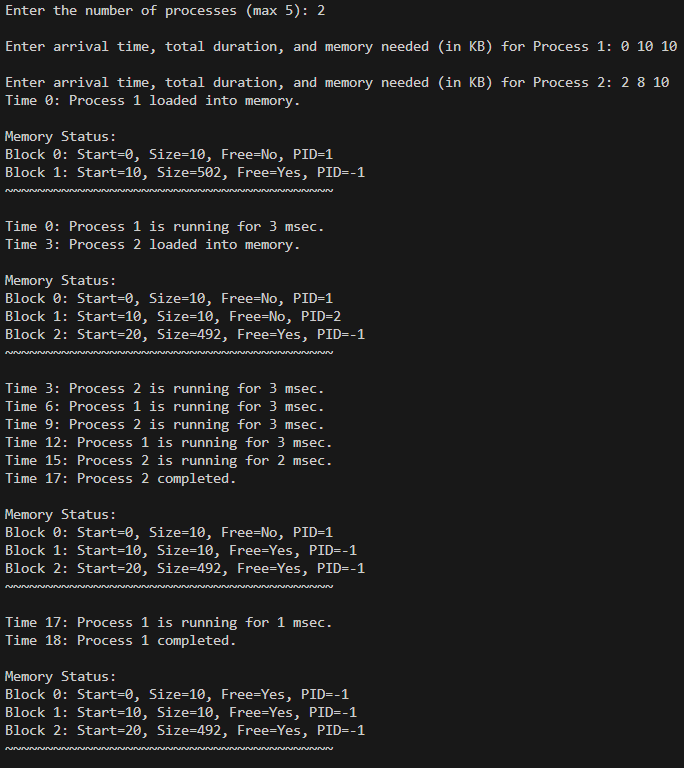
Παράδειγμα 1.

Εισάγω 2 διεργασίες, η πρώτη με χρόνο άφιξης 0, χρόνο εκτέλεσης 10 και απαιτούμενη μνήμη 10 ΚΒ και η δεύτερη με χρόνο άφιξης 2, χρόνο εκτέλεσης 8 και απαιτούμενη μνήμη 10 ΚΒ.

Process 1 (άφιξη 0 ms): Φορτώνεται στη μνήμη, εκτελείται σε 4 time slices (3 + 3 + 3 + 1 ms) και ολοκληρώνεται στα 10 ms.

Process 2 (άφιξη 2 ms): Φορτώνεται στη μνήμη μόλις η Process 1 ολοκληρωθεί, εκτελείται σε 3 time slices (3 + 3 + 2 ms) και ολοκληρώνεται στα 15 ms.

Η μνήμη αποδεσμεύεται πλήρως μετά την ολοκλήρωση και των δύο διεργασιών.



## Παράδειγμα 2.

Εισάγω 5 διεργασίες,

1. με χρόνο άφιξης 0, χρόνο εκτέλεσης 10 και απαιτούμενη μνήμη 10 ΚΒ.

2. με χρόνο άφιξης 2, χρόνο εκτέλεσης 8 και απαιτούμενη μνήμη 10 ΚΒ.

3. με χρόνο άφιξης 4, χρόνο εκτέλεσης 5 και απαιτούμενη μνήμη 10 ΚΒ.

4. με χρόνο άφιξης 6, χρόνο εκτέλεσης 15 και απαιτούμενη μνήμη 10 ΚΒ.

5. με χρόνο άφιξης 8, χρόνο εκτέλεσης 2 και απαιτούμενη μνήμη 10 ΚΒ.

Process 1: Ξεκινά από 0 ms, ολοκληρώνεται στα **21 ms**.

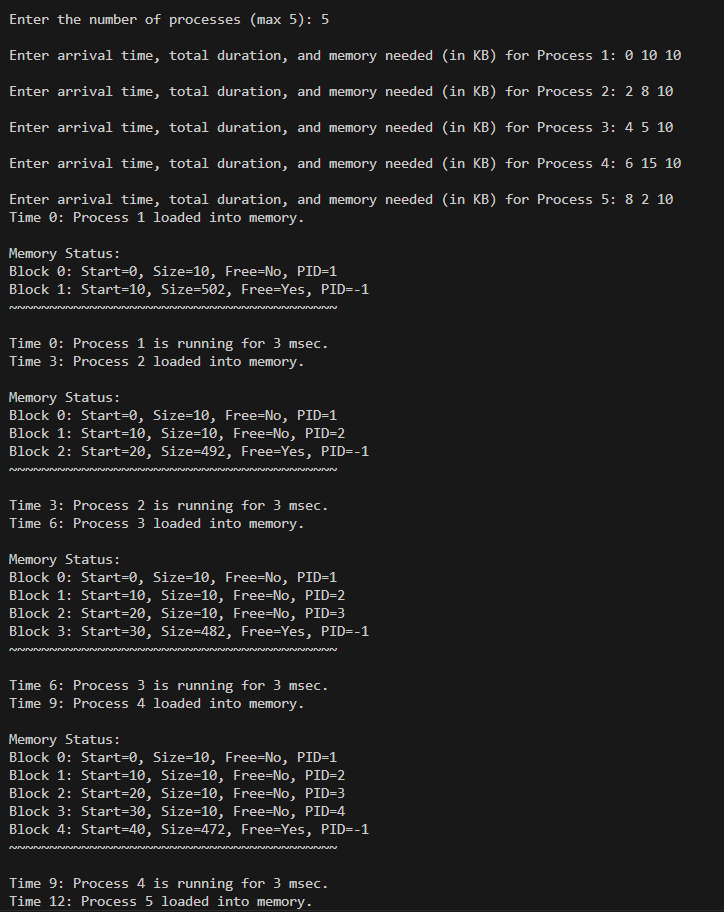
Process 2: Ξεκινά από 2 ms, ολοκληρώνεται στα **23 ms**.

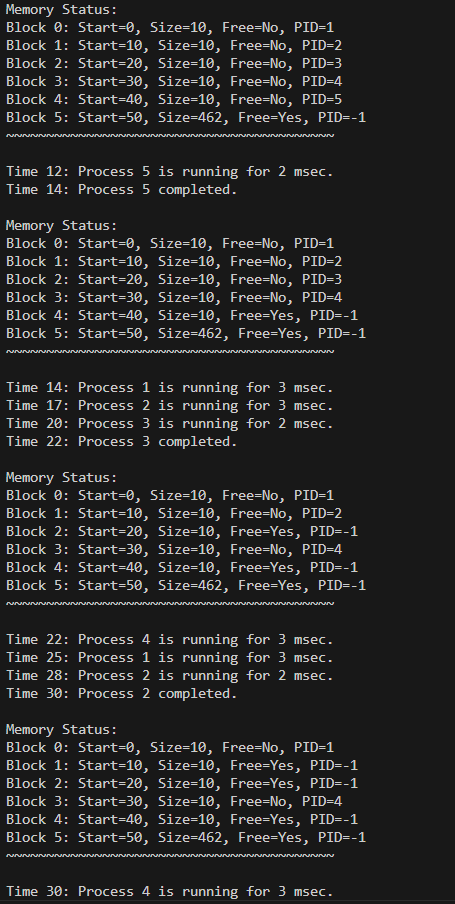
Process 3: Ξεκινά από 4 ms, ολοκληρώνεται στα **16 ms.**

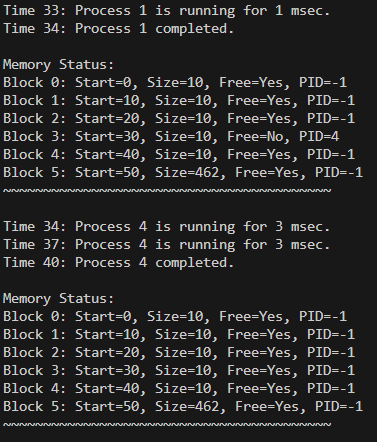
Process 4: Ξεκινά από 6 ms, ολοκληρώνεται στα **30 ms.**

Process 5: Ξεκινά από 8 ms, ολοκληρώνεται στα **10 ms**.

Η εκτέλεση ολοκληρώνεται στα **30 ms**, με όλες τις διεργασίες και τη μνήμη αποδεσμευμένες.







# 4. Χρονοπρογραμματισμός Διεργασιών

|  |  |  |  |
| --- | --- | --- | --- |
| ΔΙΕΡΓΑΣΙΑ | ΧΡΟΝΟΣ ΑΦΙΞΗΣ | ΔΙΑΡΚΕΙΑ ΕΚΤΕΛΕΣΗΣ | PID |
| Α | 0 | 6 | 3 |
| Β | 2 | 4 | 1 |
| Γ | 3 | 1 | 2 |
| Δ | 4 | 3 | 5 |
| Ε | 5 | 4 | 4 |
| Ζ | 6 | 7 | 1 |

Παρακάτω φαίνονται τα διαγράμματα Gant καθε αλγόρίθμου χρονοδρομολόγησης, μαζί με τους πίνακες υπολογισμού των μέσων τιμών των χρόνων.

α)

FCFS:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Α | Α | Α | Α | Α | Α | B | B | B | B | Γ | Δ | Δ | Δ | Ε | Ε | Ε | Ε | Ζ | Ζ | Ζ | Ζ | Ζ | Ζ | Ζ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ΔΙΕΡΓΑΣΙΑ | ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ | ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ | ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ | ΘΕΜΑΤΙΚΕΣ ΑΛΛΑΓΕΣ |
| Α | 0 | 0 | 6 | 1 |
| Β | 4 | 4 | 8 | 1 |
| Γ | 7 | 7 | 8 | 1 |
| Δ | 7 | 7 | 10 | 1 |
| Ε | 9 | 9 | 13 | 1 |
| Ζ | 12 | 12 | 19 | 1 |
| ΜΕΣΟΣ ΟΡΟΣ | 39/6=6.5 | 39/6=6.5 | 64/6=10.67 | 1 |

SJF:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A:6 |  | B:4  A:4 | Γ:1  B:4  A:3 | Δ:3  Γ:1  B:4  A:2 | Ε:4  Δ:3  Γ:1  B:4  A:1 | Ζ:7 Ε:4  Δ:3  Γ:1  B:4 | Ζ:7 Ε:4  Δ:3  B:4 |  | Ζ:7 Ε:4  B:4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Α | Α | Α | A | Α | Α | Γ | Δ | Δ | Δ | B | B | B | B | Ε | Ε | Ε | Ε | Ζ | Ζ | Ζ | Ζ | Ζ | Ζ | Ζ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ΔΙΕΡΓΑΣΙΑ | ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ | ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ | ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ | ΘΕΜΑΤΙΚΕΣ ΑΛΛΑΓΕΣ |
| Α | 0 | 0 | 6 | 1 |
| Β | 10 | 10 | 12 | 1 |
| Γ | 3 | 3 | 4 | 1 |
| Δ | 3 | 3 | 6 | 1 |
| Ε | 9 | 9 | 13 | 1 |
| Ζ | 12 | 12 | 19 | 1 |
| ΜΕΣΟΣ ΟΡΟΣ | 37/6=6.17 | 37/6=6.17 | 60/6=10 | 1 |

SRTF:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A:6 |  | B:4  A:4 | Γ:1  B:4  A:3 | Δ:3  B:4  A:3 | Ε:4  Δ:3  B:4  A:2 | Ζ:7  Ε:4  Δ:3  B:4  A:1 | Ζ:7 Ε:4  Δ:3  B:4 |  | Ζ:7 Ε:4  B:4 |  |  |  |  | Ζ:7 Ε:4 |  |  |  | Ζ:7 |  |  |  |  |  |  |
| Α | Α | Α | Γ | Α | Α | Α | Δ | Δ | Δ | B | B | B | B | Ε | Ε | Ε | Ε | Ζ | Ζ | Ζ | Ζ | Ζ | Ζ | Ζ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ΔΙΕΡΓΑΣΙΑ | ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ | ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ | ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ | ΘΕΜΑΤΙΚΕΣ ΑΛΛΑΓΕΣ |
| Α | 1 | 0 | 7 | 2 |
| Β | 8 | 8 | 12 | 1 |
| Γ | 0 | 0 | 1 | 1 |
| Δ | 3 | 3 | 6 | 1 |
| Ε | 9 | 9 | 13 | 1 |
| Ζ | 12 | 12 | 19 | 1 |
| ΜΕΣΟΣ ΟΡΟΣ | 33/6=5.5 | 32/6= 5.34 | 58/6=9.67 | 7/6=1.17 |

RR:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A:6 |  | A:4  B:4 | Γ:1  A:4  B:3 | B:2  Δ:3  Γ:1  A:4 | Ε:4  B:2  Δ:3  Γ:1  A:3 | A:2  Ζ:7 Ε:4  B:2  Δ:3  Γ:1 | A:2  Ζ:7 Ε:4  B:2  Δ:3 |  | Δ:1  A:2  Ζ:7 Ε:4  B:2 |  | Δ:1  A:2  Ζ:7 Ε:4 |  | Ε:2  Δ:1  A:2  Ζ:7 |  | Ζ:5  Ε:2  Δ:1  A:2 |  | Ζ:5  Ε:2  Δ:1 | Ζ:5  Ε:2 |  |  |  |  |  |  |
| Α | Α | B | B | Α | Α | Γ | Δ | Δ | B | B | Ε | Ε | Ζ | Ζ | A | A | Δ | Ε | Ε | Ζ | Ζ | Ζ | Ζ | Ζ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ΔΙΕΡΓΑΣΙΑ | ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ | ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ | ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ | ΘΕΜΑΤΙΚΕΣ ΑΛΛΑΓΕΣ |
| Α | 11 | 0 | 17 | 3 |
| Β | 5 | 0 | 9 | 2 |
| Γ | 3 | 3 | 4 | 1 |
| Δ | 11 | 3 | 14 | 2 |
| Ε | 11 | 6 | 15 | 2 |
| Ζ | 12 | 7 | 19 | 2 |
| ΜΕΣΟΣ ΟΡΟΣ | 53/6=8.84 | 19/6=3.17 | 78/6=13 | 12/6=2 |

γ)

Σύντομος ψευδκοκώδικας για την λειτούργια του αλγόριθμου LRTFP:

Οσο υπάρχουν διεργασίες προς εκτέλεση:

Ενημέρωσε τη λίστα των έτοιμων διεργασιών (Ready Queue).

Αν υπάρχει τρέχουσα διεργασία:

Ενημέρωσε τον χρόνο που απομένει (Remaining Time) της τρέχουσας διεργασίας.

Αν η Remaining Time της τρέχουσας διεργασίας γίνει 0:

Ολοκλήρωσε την τρέχουσα διεργασία και αφαίρεσέ την από τη Ready Queue.

Επέλεξε τη διεργασία με τον μεγαλύτερο Remaining Time από τη Ready Queue:

Αν υπάρχει ισοδυναμία, επέλεξε τη διεργασία με το μικρότερο PID.

Αν η νέα διεργασία δεν είναι η ίδια με την τρέχουσα:

Κάνε προεκχώρηση και ξεκίνησε την εκτέλεση της νέας διεργασίας.

Προχώρησε την εκτέλεση κατά 1 μονάδα χρόνου.

LRTFP:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Α:6 |  | B:4  Α:4 | Γ:1  B:3  Α:4 | Δ:3  Γ:1  B:3  Α:3 | Ε:4  Δ:3  Γ:1  B:2  Α:3 | Ζ:7  Ε:3  Δ:3  Γ:1  B:2  Α:3 |  |  |  | Ζ:3  Ε:3  Δ:3  Γ:1  B:2  Α:3 | Ζ:2  Ε:3  Δ:3  Γ:1  B:2  Α:3 | Ζ:2  Ε:3  Δ:3  Γ:1  B:2  Α:2 | Ζ:2  Ε:2  Δ:3  Γ:1  B:2  Α:2 | Ζ:2  Ε:2  Δ:2  Γ:1  B:2  Α:2 | Ζ:2  Ε:2  Δ:2  Γ:1  B:1  Α:2 | Ζ:1  Ε:2  Δ:2  Γ:1  B:1  Α:2 | Ζ:1  Ε:2  Δ:2  Γ:1  B:1  Α:1 | Ζ:1  Ε:1  Δ:2  Γ:1  B:1  Α:1 | Ζ:1  Ε:1  Δ:1  Γ:1  B:1  Α:1 | Ζ:1  Ε:1  Δ:1  Γ:1  Α:1 | Ε:1  Δ:1  Γ:1  Α:1 | Ε:1  Δ:1  Α:1 | Ε:1  Δ:1 | Δ:1 |
| Α | Α | B | Α | B | Ε | Ζ | Ζ | Ζ | Ζ | Ζ | Α | Ε | Δ | B | Ζ | Α | Ε | Δ | B | Ζ | Γ | Α | Ε | Δ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ΔΙΕΡΓΑΣΙΑ | ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ | ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ | ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ | ΘΕΜΑΤΙΚΕΣ ΑΛΛΑΓΕΣ |
| Α | 17 | 0 | 23 | 5 |
| Β | 14 | 0 | 18 | 4 |
| Γ | 18 | 18 | 19 | 1 |
| Δ | 18 | 9 | 21 | 3 |
| Ε | 14 | 0 | 19 | 4 |
| Ζ | 8 | 0 | 15 | 3 |
| ΜΕΣΟΣ ΟΡΟΣ | 89/6=14.8 | 27/6=4.5 | 115/6=19.1 | 20/6=3.33 |