Καραισκου Μαρια Ελεννη 1103077 up1103077@ac.upatras.gr

Παναγοπουλου Βασιλικη 1100651 up1100651@ac.upatras.gr

Βιτσιου Ελενη 1100502 [up1100502@ac.upatras.gr](mailto:up1100502@ac.upatras.gr)

Μποζικη Φαιδρα 1100638 up1100638@ac.upatras.gr

**1. Shell Scripting.**

**1.1 Τεκμηρίωση Κώδικα.**

Το ερώτημα 1 έχει υλοποιηθεί για να δέχεται δεδομένα από άλλα αρχεία. Για να πραγματοποιηθεί αυτό πρέπει το αρχείο να βρίσκεται στον φάκελο της άσκησης και το πρόγραμμα δέχεται τα δεδομένα μόνο με το όνομα του αρχείου, χωρίς το Path.

Στο ερώτημα 4 κατά την εκτύπωση όλων των δεδομένων, το πρόγραμμα έχει υλοποιηθεί έτσι ώστε να πατάει <enter> και όχι <space> για να συνεχιστεί η προβολή των δεδομένων.

**1.2 Περιγραφή Σχεδιασμού.**

Το πρόγραμμα αποτελείται από 4 συναρτήσεις και ένα μενού.

* Η πρωτη συναρτηση input\_data() αναφερεται στο πρωτο ερωτημα της ασκησης και υλοποιει την εισαγωγη δεδομενων ειτε από αρχειο ειτε από το πληκτρολογιο.
* Η δευτερη συναρτηση display\_passanger() αναφερεται στο δευτερο ερωτημα και πραγματοποιει την αναζητηση επιβατων. Δεχεται ως ορισμα το ID ή το ονομα ενός επιβατη και εμφανιζει τα στοιχεια του επιβατη.
* Η τριτη συναρτηση update\_field() αναφερεται στο τριτο ερωτημα και είναι υπευθυνη για την ενημερωση των στοιχειων του πινακα. Πιο συγκεκριμενα, δεχεται ως ορισμα το ID ή το ονομα ενός επιβατη και εμφανιζει την εγγραφη που υπαρχει στον συγκεκριμενο επιβατη. Ο χρηστης πρεπει να συμπληρωσει το πεδιο που θελει να αλλαξει και την νεα τιμη του πεδιου και στην συνεχεια εκτυπωνεται η νεα εγγραφη του επιβατη.
* Η τεταρτη συναρτηση display\_file() απευθυνεται στο τεταρτο ερωτημα και εμφανιζει ολες τις εγγραφες του αρχειου passangers.csv. Η συναρτηση υπολογιζει τις γραμμες τις οθονης του υπολογιστη και όταν η οθονη γεμισει με εγγραφες εκτυπωνει ένα μηνυμα οπου ο χρηστης πρεπει να επιλεξει <enter> ή <e> για συνεχεια εκτυπωσης εγγραφης ή εξοδο αντιστοιχα.
* Το μενου είναι υπευθυνο για να πραγματοποιησει την ενεργεια που θελει να υλοποιησει ο χρηστης. Αναλογα με την ενεργεια του χρηστη επιλεγεται να κληθει η αντιστοιχη συναρτηση.

**1.3 Παραδειγματα Λειτουργιας.**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εισαγωγη Δεδομενων από το πληκτρολογιο.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Αναζητηση επιβατη.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Επεξεργασια εγγραφης.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εμφανιση εγγραφων μεχρι να γεμισει η οθονη.

**2. Συγχρονισμός Διεργασιών και Σημαφόροι.**

**2.1 Τεκμιριωση Ασκησης.**

Το ερωτημα 2 εχει υλοποιηθει για να διαχειριζεται επιβατες σε λεμβους συγκεκριμενης χωριτικοτητας. Ο κωδικας συμπεριλαμβανει 3 αρχεια <ipc\_utils.h> , <passenger.c>, <launch.c>.

**2.2 Περιγραφη Σχεδιασμου.**

Το αρχειο <ipc\_utils.h> είναι υπευθυνο για την ρυθμιση των σημαιοφορων.

* Πιο συγκεκριμενα, περιεχει την συναρτηση lifeboat\_init() στην οποια αρχικοποιειται ο σημαιοφορος boarding\_sem με αρχικη τιμη μεγιστη δυνατη χωριτηκοτητα και αρχικοποιειται ο δυαδικος σημαιοφορος mutex για τον συγχρονισμο.
* Η συναρτηση lifeboat\_destroy() αποδεσμευει τους σημαιοφορους.
* Η συναρτηση board\_lifeboat() είναι υπευθυνη για την επιβιβαση ενός επιβατη. Αναλυτικοτερα, ελεγχει αν υπαρχουν διαθεσιμες θεσεις στις λεμβους και μετραει τους επιβατες που επιβιβαζονται στην λεμβο.
* Η συναρτηση disembark\_lifeboat() απελευθερωνει θεσεις στις λεμβους κάθε φορα που αποβιβαζεται ενας επιβατης.

Το αρχειο <passenger.c> είναι υπευθυνο για να επεξεργαζεται επιβατες που προσπαθουν να επιβιβαστουν στις λεμβους.

* Περιεχει την συναρτηση passenger() η οποια καλει τις συναρτησεις board\_lifeboat() και disembark\_lifeboat() του αρχειου <ipc\_utils.h> .

Το αρχειο <launch.c> διαβαζει το πληθος των επιβαινοντων, των λεμβων και την χωρητικοτητα των λεμβων και ρυθμιζει τις θεσεις στις λεμβους.

* Η συναρτηση passenger\_thread() καλει την συναρτηση passenger() του αρχειου <passenger.c> χρησιμοποιωντας την καταλληλη λεμβο .
* Η συναρτηση main() δεσμευει μνημη για τον πινακα LifeBoat που αντιστοιχει στον αριθμο των λεμβων και δημιουργει τις λεμβους. Στην συνεχεια, δημιουργει πινακα νηματων για τους επιβατες δευσμευοντας μνημη για τα ορισματα του νηματος. Τελος, περιμενει όλα τα νηματα να ολοκληρωθουν , καταστρεφει τους σημαιοφορους κάθε λεμβου και αποδεσμευει την μνημη για τις λεμβους.

**2.3 Παραδείγματα υλοποίησης.**

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα**

Παραδειγμα εκτελεσης προγραμματος

με μια λεμβο 2 θεσεων και 5 επιβατες.

**3. Χρονοπρογραμματισμός Διεργασιών και Διαχείριση Μνήμης**

**3.1 Τεκμηρίωση Κώδικα:**

Για την υλοποίηση του κώδικα δημιούργησα 4 συναρτήσεις, οι οποίες εκτελούν τις εξής διεργασίες: Αρχικοποίηση του Memory Block, δέσμευση μνήμης, αποδέσμευση μνήμης και Round Robin.

* initialize\_memory(); Συνάρτηση αρχικοποίησης του Memory Block. Θέτει όλα τα στοιχεία του σε 0 (Η μνήμη ξεκινάει ελεύθερη). Χρησιμοποιείται δυναμική διαχείριση μνήμης.
* allocate\_memory(int pid, int memory\_needed); Συνάρτηση δέσμευσης μνήμης. Καταχωρείται μνήμη για μία διεργασία με βάση το αναγκαίο μέγεθός της. Διαχωρίζει το block αν είναι μεγαλύτερο από το απαιτούμενο μέγεθος.
* deallocate\_memory(int pid); Συνάρτηση αποδέσμευσης μνήμης. Αντίστοιχα με την προηγούμενη συνάρτηση, συγχωνεύει τα ελεύθερα block.
* simulate(); Συνάρτηση υλοποίησης του αλγορίθμου Round Robin. Οι διεργασίες φορτώνονται στην μνήμη με σειρά άφιξης. Έπειτα εκτελούνται κυκλικά, σύμφωνα με τον Round Robin. Όταν ολοκληρώνεται μία διεργασία, ο χώρος που καταλάμβανε αποδεσμεύεται. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να ολοκληρωθούν όλες οι διεργασίες.

**3.2 Key Points.**

* Αν κάποια διεργασία ξεπερνάει το όριο του χώρου της μνήμης τότε δεν φορτώνεται στην μνήμη.
* Μόνο οι διεργασίες που έχουν φορτωθεί στην μνήμη μπορούν να εκτελεστούν.
* Αν κάποια διεργασία χρειάζεται περισσότερο χώρο από τον διαθέσιμο, η συνάρτηση allocate\_memory() επιστρέφει false, οπότε η διεργασία δεν φορτώνεται στην μνήμη και παραμένει σε αναμονή στην simulate() έως ότου απελευθερωθεί αρκετός χώρος. Έπειτα στην simulate() συνεχίζει να εξετάζεται η επόμενη διεργασία.
* Όταν αποδεσμεύεται μνήμη, σύμφωνα με την deallocate\_memory(), τα γειτονικά block συγχωνεύονται.

**3.3 Παραδειγματα Λειτουργιας.**

* Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κατάλογος, γραμματοσειρά

  Περιγραφή που δημιουργήθηκε αυτόματαTest 1:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα

Η 1η διεργασία δεν φορτώνεται καθώς ξεπερνάει τον διαθέσιμο χώρο μνήμης.

Η 3η διεργασία τρέχει για 4 χρόνους καθώς η επόμενη διεργασία δεν έχει φορτωθεί ακόμα.

Όλες οι διεργασίες φορτώνονται κανονικά στους χρόνους τους με την σειρά.

* Τest 2:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κατάλογος, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα

Οι διεργασίες φορτώνονται κανονικά στους χρόνους τους, εφόσον υπάρχει διαθέσιμος χώρος.

Έπειτα τρέχουν κυκλικά ανά 3 χρόνους σύμφωνα με την Round Robin (time quantum = 3).

* Τest 3:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, κατάλογος

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κατάλογος

Περιγραφή που δημιουργήθηκε αυτόματα

Οι διεργασίες φορτώνονται με την σωστή σειρά στην μνήμη και εκτελούνται σύμφωνα με τον Round Robin.

**4. Χρονοπρογραμματισμός Διεργασιών**

**α)**

Στην 4η Άσκηση ερώτημα α μας ζητήθηκε σχεδιασμός των διαγραμμάτων Gantt.

1. Στον πρώτο αλγόριθμο χρονοδρομολόγησης (FCFS), οι διεργασίες εκτελούνται με τη σειρά με την οποία φτάνουν στην ουρά διεργασιών. Κοιτώντας λοιπόν την ουρά, παρατηρούμε ότι την χρονική στιγμή 0 φθάνει η διεργασία Α η οποία και εκτελείτε μεχρι να ολοκληρωθει, καθως ο αλγοριθμος δεν ειναι προεκχωρητικος. Μεχρι να ολοκληρωθει η Α, στην ουρα εχουν φτασει η Β (ΧΑ=2), Γ (ΧΑ=3), Δ (ΧΑ=4), Ε (ΧΑ=5) και Ζ (ΧΑ=6), σε αυτη την σειρα οποτε και εκτελεουνται με την σειρα αυτη. Ο χρονος ολοκληρωσεις ειναι ισος με το άθροισμα όλων των ΧΕ καθώς δεν εχουμε καποια στιγμη αδράνειας στην ΚΜΕ.

1. Στον αλγοριθμο SJF, η ΚΜΕ εκετελεί την διεργασία που έχει τον μικρότερο χρόνο εκτέλεσης πρωτη. Ουτε ο αλγοριθμος SJF ειναι προεκχωρητικος, επομένως οταν φτανει στην ουρα η Α εκτελειτε μεχρι να τελειωσει. Μεχρι να ολοκληρωθει η Α, εχουν φτασει ολες οι υπολοιπες διεργασιες στην ουρα διεργασιων με την σειρα που αναφεραμε και απο πανω. Η σειρα εκτελεσης των διεργασιων εξαρτατε απο τον χρονο εκτελεσης της καθε μιας( Β=4, Γ=1, Δ=3, Ε=5, Ζ=7) και επομένως εχουμε Γ, Δ, Β, Ε, Ζ σειρα εκτελεσης των διεργασιων.Ο χρονος ολοκληρωσεις ειναι ιδιος με πριν καθως δεν εχουμε στιγμες αδρανειας της ΚΜΕ.

1. Στον αλγοριθμο SRTF, η ΚΜΕ εκτελεί την διεργασία που εχει τον μικροτερο εναπομιναντα χρονο. Επειδη ο συγκεκριμενως αλγοριθμος ειναι προεκχωρητικος, καθε φορα που μια διεργασια καταφθανει στην ουρα διεργασιων, ελεγχει αν ο χρονος εκτελεσης την διεργασιας που εφτασε ειναι μικρότερος απο αυτον της διεργασιας που χρησιμοποιηουσε την ΚΜΕ. Σε περιπτωση που εχουν τον ιδιο χρονο, τοτε εκτελει την διεργασια που ειχε ηδη την ΚΜΕ. Οπως φαινεται και στο διαγραμμα Gantt η διεργασια Α φτανει στην ουρα και εκτελειτε μεχρι την χρονικη στιγμη 2, οπου φτανει στην ουρα διεργασιων η Β. Στην Α απομενει ο ιδιος χρονος εκτελεσεις που εχει και η Β(6 χρονος εκτελεσης μειον 2 ο χρονος που εκτελεστηκε η Α), επομενως η Α καταλαμβανει την ΚΜΕ μεχρι την χρονικη στιγμη 3 που φτάνει η διεργασια Γ. Επειδη η Γ εχει μικροτερο χρονο εκτελεσης απο την Α , την ΚΜΕ παιρνει η Γ. Η εκτελεση των υπολοιπων διεργασιων συεχιζεται αντιστοιχα, μεχρι την χρονικη στιγμη 26 που θα εχουν εκτελεστει ολες οι διεργασιες.

1. Ο αλγοριθμος Round Robin, κατανεμει σε κάθε διεργασία ίσο μερίδιο του χρόνου της ΚΜΕ. Οι εργασίες τοποθετούνται σε μια κυκλική ουρά και όταν λήξει ο εκχωρημένος χρόνος μιας διεργασίας, μεταφέρεται στο τέλος της ουράς ενώ η επόμενη εργασία λαμβάνεται από το μπροστινό μέρος. Καθε φορα που μια διεργασία αφήνει την ΚΜΕ και την ίδια χρονική στιγμή καταφθάνει μια νέα διεργασία στο σύστημα, τότε η διεργασία που αφήνει την ΚΜΕ εισέρχεται πρώτη στην ουρά των διεργασιών. Επειδη η ΚΜΕ δινει σε καθε διεργασία 2 κβαντα χρονου η Α που φτανει πρωτη στην ουρα εκτελειτε μεχρι την χρονικη στιγμη 2, οπου φτανει στην ουρα η Β. Η Β φτανει στην ουρα την ιδια χρονικη στιγμη που η Α αφεινει την KME, γιαυτο θεωρουμε οτι η A φτανει πρωτη στην ουρα και επομενως ειναι η επομενη που καταλαβμανει την ΚΜΕ. Την χρονικη στιγμη 3 φτανει στην ουρα και η Γ. Την χρονικη στιγμη 4 η Α αφηνει την ΚΜΕ και ταυτοχρωνα στην ουρα φτανει και η Δ. Η Β καταλαμβανει την ΚΜΕ και η ουρα των διεργασιων ειναι Γ, Α, Δ. Η Γ ολοκληρωνετε την στιγμη 5 και φτανει στην ουρα η Ε. Η εκτελεση των υπολοιπων διεργασιων συνεχιζεται αντιστοιχα.

**β)**

Στο ερωτημα β ζητειτε για καθε αλγοριθμο ο μεσος χρονος αναμονης, αποκρισης, ολοκληρωσης και το πληθος των θεματικων εναλλαγων των διεργασιων.

1. Ο μεσος χρονος ολοκληρωσης ειναι ισος με την στιγμη ολοκληρωσης της καθε διεργασιας μειον την στιγμη που αυτη εφτασε στην ουρα των διεργασιων και όλο δια τον αριθμο των διεργασιων. Για παράδειγμα, για τον αλγόριθμο FCFS έχουμε για την Α χρόνος ολοκλήρωσης 6 και χρόνος άφιξης 0, άρα χρόνο ολοκληρωσης 6. Για το Β έχουμε χρονος ολοκληρωσης 10 και χρόνος άφιξης 2 άρα χρόνος ολοκλήρωσης 8. Ομοίως βρίσκουμε και για τις υπόλοιπες διεργασίες αφού τα αθροισουμε διαιρούμε με τον αριθμο των διεργασιων και βρίσκουμε τον μέσο χρόνο ολοκληρωσης.

1. Ο μεσος χρονος αναμονης ειναι ισος με τον χρόνο ολοκληρωσης μειον τον χρόνο εκτέλεσης και όλο δια τον αριθμο των διεργασιων. Για παράδειγμα για τον αλγόριθμο FCFS έχουμε για το Α χρόνο ολοκληρωσης 6 και χρόνο εκτέλεσης 6 άρα χρόνος αναμονής 0. Για το Β έχουμε χρονο ολοκληρωσης 8 και χρόνο εκτέλεσης 4 άρα χρονος αναμονής 4. Ομοίως βρίσκω και αθροιζω τους υπόλοιπους χρόνους αναμονης και έπειτα διερω με των αριθμο των διεργασιων (6).

1. Ο μεσος χρονος αποκρισης είναι ισος με το άθροισμα των χρόνων που η κάθε διεργασια παίρνει για πρώτη φορά την ΚΜΕ μειον τον χρόνο άφιξης της διεργασιας και όλο δια τον αριθμο των διεργασιων. Για παράδειγμα στον αλγόριθμο FCFS η Α καταλαμβάνει την ΚΜΕ την στιγμη 0 με χρόνο άφιξης 0 επομένως ο χρονος αποκρισης είναι 0. Η Β καταλαμβάνει την ΚΜΕ την χρονική στιγμή 6 και έχει χρόνο άφιξης 2 επομένως ο χρονος αποκρισης είναι 4. Παρόμοια βρίσκουμε και αθροιζουμε τον χρόνο αποκρισης για τις υπόλοιπες διεργασίες και διαιρούμε με το πληθος των διεργασιων.

1. Το πληθος των θεματικων εναλλαγων των διεργασιων ειναι ο αριθμός των εναλλαγων διεργασιων που γίνονται στην ΚΜΕ. Κάθε φορά που μια διεργασια αφήνει την ΚΜΕ και την θεση της παίρνει μια άλλη, έχουμε μια εναλλαγή. Για παράδειγμα στον αλγόριθμο FCFS παρατηρώ πως έχουμε 5 θεματικές εναλλαγές όσες δηλαδή οι εναλλαγές διεργασιων στην ΚΜΕ ( από Α στην Β , από Β στην Γ, από Γ στην Δ, από Δ στην Ε και από Ε στην Ζ).