

Λειτουργικά Συστήματα

2024 - 2025

1η Εργαστηριακή Άσκηση (Project)

Άσκηση 1 Shell Scripting

Σε κάθε λειτουργία χρειάζεται η διαδρομή για το αρχείο passenger_data.csv. Για κάθε μία ορίσαμε την διαδρομή και προσθέσαμε έλεγχο αν υπάρχει με κατάλληλο μήνυμα στον χρήστη.

Για την καλύτερη διαχείριση του shell script για κάθε λειτουργία δημιουργήσαμε συνάρτηση και χρησιμοποιήσαμε flags.

```
# Function for displaying usage
display_usage() {
    echo "Usage: $0 [--insert | --search | --update | --help]"
    echo "  --insert      Insert data into passengers.csv"
    echo "  --search      Search for a passenger in passengers.csv"
    echo "  --update      Update passenger data in passengers.csv"
    echo "  --display      Display passenger data "
    echo "  --reports      Generate reports"
    echo "  --help        Display this help message"
    exit 1
}

# Main script logic with flags
if [[ $# -eq 0 ]]; then
    display_usage
fi

case "$1" in
    --insert)
        insert_data
        ;;
    --search)
        search_passenger
        ;;
    --update)
        update_passenger
        ;;
    --display)
        display_file
        ;;
    --reports)
        generate_reports
        ;;
    --help)
        display_usage
        ;;
    *)
        echo "Error: Invalid flag"
        display_usage
        ;;
esac
```

[1] Εισαγωγή δεδομένων στην εφαρμογή: ./processes_ipc.sh --insert

Ζητείται από τον χρήστη να επιλέξει τον τρόπο εισαγωγής επιβατών (πληκτρολόγιο ή αρχείο). Αν ο χρήστης πατήσει Enter (δηλαδή το file_path είναι κενό), η εισαγωγή γίνεται μέσω πληκτρολογίου. Αν δοθεί διαδρομή αρχείου, η συνάρτηση προσπαθεί να διαβάσει τα δεδομένα από το συγκεκριμένο αρχείο.

Εισαγωγή μέσω πληκτρολογίου : Για κάθε γραμμή ελέγχεται η μορφή με regex.(if [[! "\$line" =~ ^([^\,]+,[^\,]+,[0-9]+,[^\,]+,(Passenger|Crew),(Yes|No)\$]];). Αν είναι έγκυρη προστίθεται στο αρχείο (echo "\$line" >> passenger_data.csv), διαφορετικά εμφανίζεται μήνυμα σφάλματος. Ο χρήστης τερματίζει την εισαγωγή πατώντας Ctrl+D.

Εισαγωγή διαδρομής: Στην αρχή, υπάρχει έλεγχος αν το αρχείο υπάρχει. Το αρχείο διαβάζεται γραμμή-γραμμή (while IFS= read -r line; do) και ελέγχεται με το ίδιο regex.

Δομή regex: ^([^\,]+,[^\,]+,[0-9]+,[^\,]+,(Passenger|Crew),(Yes|No)\$

Το πρώτο, δεύτερο και τέταρτο πεδίο πρέπει να μην είναι κενά, το τρίτο πεδίο πρέπει να είναι αριθμός, το πέμπτο πεδίο πρέπει υποχρεωτικά να είναι Passenger ή Crew και το τελευταίο Yes ή No.

Παρακάτω υπάρχει παράδειγμα: 1. εισαγωγή μέσω πληκτρολογίου 2. εισαγωγή αρχείου 3. screenshot του ανενωμένου αρχείου με τις επιτυχημένες προσπάθειες εισαγωγής επιβατών

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh -  
-insert  
Enter the full path of the file or press Enter to input data manually:  
  
Enter data in the format: [code],[fullname],[age],[country],[status (Passenger/Crew)],  
[rescued (Yes/No)]  
Press Ctrl+D when finished.  
1904,gigi,4,greece,Passenger,Yes  
Data successfully inserted into passenger_data.csv.  
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$
```

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh -  
-insert  
Enter the full path of the file or press Enter to input data manually:  
/home/gg/Documents/shell_workspace/passengers_data_example.csv  
Invalid format in file: code,fullname,age,country,status,rescued  
Data successfully inserted into passenger_data.csv.  
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$
```

```
1275 1274;Yang Yang;67;Croatia;Passenger;yes  
1276 1275;Lopez Cesar;68;Bosnia and Herzegovina;Passenger;yes  
1277 1276;Zhang Yang;69;Moldova;Crew;no  
1278 1277;Hussain Tariq;70;Lithuania;Passenger;Yes  
1279 1278;Dia Ibrahima;71;Albania;Passenger;Yes  
1280 1279;Abouba Zakari;72;Slovenia;Passenger;yes  
1281 1280;Zhu Hao;73;Latvia;Passenger;yes  
1282 1904,gigi,4,greece,Passenger,Yes  
1283 102,Jane Smith,34,Canada,Crew,No  
1284 102,Jane Smith,34,Canada,Crew,No  
1285 101,John Doe,30,USA,Passenger,Yes  
1286 1904,gigi,4,greece,Passenger,Yes  
1287 9005,Rosie Rosie,30,USA,Passenger,Yes  
1288 9006,Reb Kou,25,Japan,Crew,No  
1289 9007,Emily Brown,22,UK,Passenger,Yes  
1290
```

Σημείωση: το αρχείο passengers_data_example.csv, δημιουργήθηκε με την βοήθεια ενός σύντομου script σε python.

Πρόβλημα που παρουσιάστηκε και δεν λύθηκε: όπως φαίνεται και στην δεύτερη εικόνα, παρόλου που τα δεδομένα του αρχείου εισάγονται επιτυχώς εμφανίζεται μήνυμα λάθους.

[2]Προβολή στοιχείων επιβαίνοντα από το αρχείο: ./processes_ipc.sh --search

Κατά την εκτέλεση του προγράμματος εμφανίζεται μήνυμα στον χρήστη για να εισάγει τα στοιχεία του επιβάτη που αναζητά.

Η αναζήτηση στο αρχείο γίνεται με την συνάρτηση grep.

Με την χρήση -i, επιτυγχάνεται case-insensitive.

Καλύπτεται το ενδεχόμενο αποτυχημένης αναζήτησης με ενημέρωση του χρήστη.

```
([[ -z "$result" ]]:)
```

Παρακάτω υπάρχει παράδειγμα: 1. αναζήτηση με fullname 2. αναζήτηση με name 3. αποτυχής αναζήτηση.

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh --search
Enter the name or surname of the passenger to search:
basov irina
Passenger(s) found:
24;Basov Irina;43;Norway;Passenger;Yes
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$
```

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh --search
Enter the name or surname of the passenger to search:
fischer
Passenger(s) found:
78;Fischer Ernst;11;Estonia;Passenger;Yes
118;Fischer Otto;51;Slovenia;Passenger;no
449;Fischer Fritz;78;Switzerland;Passenger;yes
682;Fischer Walter;83;Montenegro;Crew;no
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$
```

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh --search
Enter the name or surname of the passenger to search:
gghjkhgf jkhg
No matching passenger found.
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$
```

[3]Αλλαγή στοιχείων επιβαίνοντα: `./processes_ipc.sh --update`

Η αναζήτηση των επιβάτων γίνεται με τον ίδιο τρόπο όπως στην προηγούμενη λειτουργία.

Αν εντοπιστεί ο επιβάτης, το σύστημα παροτρύνει τον χρήστη να εισάγει το πεδίο προς ενημέρωση. Το πεδίο `field_value`, που είναι το όρισμα που δίνει ο χρήστης, χωρίζεται σε δύο μέρη: `field` → το όνομα του πεδίου και `new value` → τη νέα τιμή.

Ζητείται από τον χρήστη να εισάγει το πεδίο και τη νέα τιμή διαχωρισμένα με “;”.

```
(IFS=";" read field new_value <<< "$field_value")
```

Αν το `field` είναι `record` ενημερώνεται όλη η γραμμή. Με τη χρήση του `sed`, αναζητείται η γραμμή που περιέχει το `search_query` και αντικαθίσταται. (`sed -i "/$search_query/c$new_value" "$FILE_PATH"`)

Στην περίπτωση που ενημερώνεται συγκεκριμένο πεδίο, για κάθε στήλη έχει δωθεί κατάλληλο όνομα ανάλογα με το πεδίο και ενημερώνεται το πεδίο που δόθηκε (`case "$field"`).

Η ενημέρωση του πεδίου γίνεται με τη χρήση `awk`. Ως διαχωριστής των πεδίων ορίζεται το “;”.

Για την εμφάνιση των παλιών και νέων στοιχείων χρησιμοποιήθηκαν εντολές `echo` με τα κατάλληλα ορίσματα.

Παρακάτω υπάρχει παράδειγμα : 1. αλλαγή ενός πεδίου 2. ενημέρωση στοιχείων με χρήση `record`

```

No matching passenger found.
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh -
-update
Enter the code, name, or surname of the passenger to update:
1199
Passenger found: 1199;Wang Jun;68;Malta;Crew;yes
Enter the field to update (e.g., fullname, age, country, status, rescued, or rec
ord):
rescued:no
Updating field 'rescued' to 'no'...
Passenger updated:
Old: 1199;Wang Jun;68;Malta;Crew;yes
New: 1199;Wang Jun;68;Malta;Crew;no

```

```

h
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh -
-update
Enter the code, name, or surname of the passenger to update:
1200
Passenger found: 1200;Zhu Zheng;69;Iceland;Passenger;Yes
Enter the field to update (e.g., fullname, age, country, status, rescued, or rec
ord):
record:1200;gigi;24;netherlands;Passenger;no
Updating entire record...
Passenger updated:
Old: 1200;Zhu Zheng;69;Iceland;Passenger;Yes
New: 1200;gigi;24;netherlands;Passenger;no
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$

```

[4] Προβολή αρχείου:./processes_ipc.sh –display

Το πρόγραμμα ξεκινάει με έναν βρόχο αναμονής του χρήστη. Όπου εμφανίζει μήνυμα που καθοδηγεί τον χρήστη.

Η διαχείριση εισόδου χρήστη γίνεται με την εντολή read.

Το read χρησιμοποιείται με τις επιλογές -n1 (διαβάζει μόνο έναν χαρακτήρα από το χρήστη) και -s (silent, η είσοδος δεν εμφανίζεται στην οθόνη).

Υπάρχουν ελέγχοι της μεταβλητής key (η μεταβλητή της read), ώστε να διαπιστωθεί αν είναι κενή για να ξεκινήσει η προβολή με less. Κενή μπορεί να είναι μόνο αν ο χρήστης έχει εισάγει enter

Με την εντολή less επιτρέπεται η διαδραστική προβολή του αρχείου. Η less διαβάζει το αρχείο που υποδεικνύεται από τη διαδρομή και εμφανίζει το περιεχόμενο στην οθόνη του χρήστη (less "\$FILE_PATH").

Δυνατότητες less: I) SPACE: Προχωρά στην επόμενη σελίδα.

ii) b: Προχωρά κατά μία γραμμή.

Iii) q: Τερματίζει την προβολή και επιστρέφει στο τερματικό.

iv) G: Μεταβαίνει στο τέλος του αρχείου.

v) g: Μεταβαίνει στην αρχή του αρχείου.

Η less φορτώνει μόνο τις γραμμές που είναι απαραίτητες για την εμφάνιση στην οθόνη, αντί να φορτώσει όλο το αρχείο στη μνήμη. Αυτό τη καθιστά ιδανική για μεγάλα αρχεία.

Παρακάτω υπάρχει παράδειγμα εκτέλεσης του προγράμματος.

```
Press Enter to start viewing the file with less or q to exit.  
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh --display  
Press Enter to start viewing the file with less or q to exit.
```

```
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;Znaïmer 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  
28;Goma Ina;47;Bosnia and Herzegovina;Passenger;yes  
29;Pham Anh;48;Moldova;Passenger;yes  
30;Liu Yang;49;Lithuania;Passenger;no  
31;Balmaceda Patricio;50;Albania;Passenger;yes  
32;Al-Mukhaini Isla;51;Slovenia;Passenger;yes  
33;Giannopoulos Aris;52;Latvia;Crew;yes  
34;Taylor Robert;53;North Macedonia;Passenger;Yes  
35;Rojas Consuelo;54;Estonia;Passenger;no  
36;Maszaros Janos;55;Luxembourg;Passenger;yes  
37;Reyes Antonio;46;Montenegro;Passenger;yes  
38;Svensson Ernst;47;Malta;Passenger;yes  
39;Rossi Mauro;48;Iceland;Passenger;yes  
40;Radafison Nirina;49;Andorra;Passenger;yes  
41;Gauchan Crown Prince;50;Liechtenstein;Passenger;no  
42;Dobrunov Aleksandr;51;Monaco;Passenger;yes  
43;Johnson Carl;52;San Marino;Passenger;yes  
44;Li Jun;53;Russia;Crew;no  
45;Garbi Antonis;54;Germany;Passenger;Yes  
:
```

Πρόβλημα που παρουσιάστηκε και δεν λύθηκε: σύμφωνα με την συμπεριφορά του read για Ctrl+D αποτυγχάνει και επιστρέφει μηδενική τιμή (τερματίζεται). Παρόλο που έχουμε έλεγχο για το read :
if ! read -n1 -s key; then exit 0 fi, ο οποίος ελέγχει αν η εντολή απέτυχε (ο χρήστης πάτησε Ctrl+D) δεν φαίνεται να λειτουργεί.

[5] Δημιουργία αναφορών: `./processes_ipc.sh --reports`

Εύρεση και προβολή των ηλικιακών ομάδων των επιβατών.

Χρησιμοποιήθηκε η εντολή `awk` για να κατηγοριοποιήσει τους επιβάτες σε ηλικιακές ομάδες. Ο αριθμός των επιβατών αποθηκεύεται σε ένα λεξικό `age_group` (δυναμική αποθήκευση λόγω του `awk`) και τέλος τα δεδομένα γράφονται στο αρχείο `ages.txt`.

`-F";"` → ορίζει το διαχωριστικό πεδίων σε ;

`$3` → υποδεικνύει την στήλη που περιέχει την ηλικία του κάθε επιβάτη

`age_group` → Ο πίνακας `age_group` χρησιμοποιεί τις ηλικιακές ομάδες ως κλειδιά. Η τιμή κάθε κλειδιού αντιπροσωπεύει τον αριθμό των επιβατών σε αυτή την ομάδα. Χρησιμοποιώντας έλεγχο ηλικίας ανά `group` αυξάνεται κάθε φορά κατά 1 το κλειδί όπου βρίσκεται αντιστοιχία. Το μπλοκ `END` αποτελείται από έναν βρόχο (`for (group in age_group)`), ο οποίος διατρέχει όλα τα κλειδιά του λεξικού. Στη συνέχεια η εντολή `print group, age_group[group]` εμφανίζει το όνομα του ηλικιακού `group` (π.χ. 0-18) και των αριθμό των επιβατών.

Υπολογισμός του ποσοστού των συμμετεχόντων στη διάσωση για κάθε ηλικιακή ομάδα.

Χρησιμοποιήθηκε η εντολή `awk` σε συνδυασμό με το λεξικό `rescued` για την καταγραφή των διασωθέντων.

Ο υπολογισμός του ποσοστού γίνεται με τον τύπο: $(\text{rescued}[\text{group}] / \text{age_group}[\text{group}]) * 100$.

`tolower($6)` → μετατρέπει τη στήλη `$6` σε πεζά γράμματα, για να αναγνωρίζεται σωστά το `Yes/yes`.

`Rescued` → Ο πίνακας `rescued` χρησιμοποιεί τις ηλικιακές ομάδες ως κλειδιά και οι τιμές του αντιπροσωπεύουν τον αριθμό των διασωθέντων επιβατών για κάθε ηλικιακή ομάδα. Υπάρχουν δύο έλεγχοι: έλεγχος ηλικίας με σκοπό ο κάθε επιβάτης να καταχωρωθεί στην ανάλογη ομάδα και έλεγχος του πεδίου `rescued` με σκοπό την άυξηση του ανάλογου μετρητή της ηλικιακής ομάδας (π.χ. `rescued["0-18"]++`).

Υπολογισμός μέσης ηλικίας ανά κατηγορία επιβατών.

Υπολογίζεται η μέση ηλικία ανά κατηγορία με βάση το όρισμα της στήλης `$3` (ηλικία) και της `$5` (κατηγορία)

Οι ηλικίες προστίθενται στις μεταβλητές `total_passenger` και `total_crew`, οι οποίες λειτουργούν σαν λεξικά. Ο αριθμός των μελών κάθε κατηγορίας μετρείται σε `count_passenger` και `count_crew`. Στο τέλος, η μέση ηλικία υπολογίζεται διαιρώντας το άθροισμα ηλικιών με τον αριθμό των μελών.

Φιλτράρισμα των δεδομένων ώστε να δημιουργηθεί ένα νέο αρχείο που θα περιέχει μόνο τους «διασωθέντες».

Χρησιμοποιήθηκε η εντολή `grep` για να φιλτράρει τις γραμμές του αρχείου που τελειώνουν με `“;yes”`.. Προστέθηκε στην εντολή το στοιχείο `-i` για να επιτύχουμε `case insensitive`.

Για κάθε ζητούμενο δημιουργείται το ανάλογο txt αρχείο (π.χ. passenger_data.csv > avg.txt)

Παράδειγμα εκτέλεσης της συνάρτησης.

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$ ./processes_ipc.sh -
-reports
Generating age groups report...
Calculating rescue percentages by age group...
Calculating average age by category...
Filtering rescued passengers...
Reports generated successfully:
- Age groups: ages.txt
- Rescue percentages: percentages.txt
- Average age by category: avg.txt
- Rescued passengers: rescued.txt
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/shell_workspace$
```

Άσκηση 2 Συγχρονισμός Διεργασιών και Σημαφόροι

Το πρόγραμμα επιλύει το πρόβλημα της διαχείρισης επιβατών και λέμβων χρησιμοποιώντας νήματα (threads) και σημαφόρους (semaphores) για συγχρονισμό.

α) Αρχείο ipc_utils.h

Λειτουργεί ως header file, οι συναρτήσεις που περιέχει χρησιμοποιούνται στα άλλα δύο αρχεία.

Περιλαμβάνει συναρτήσεις για τη δημιουργία και διαχείριση σημαφόρων, οι οποίες είναι απαραίτητες για τον συγχρονισμό των νημάτων.

create_semaphore : δημιουργεί έναν σημαφόρο με το όνομα και την αρχική τιμή που του δίνεται.

destroy_semaphore: κλείνει και διαγράφει έναν σημαφόρο, απαραίτητη για την αποδέσμευση πόρων.

semaphore_down: μειώνει την τιμή ενός σημαφόρου. Χρησιμοποιείται για να περιμένει ένα νήμα έως ότου γίνει διαθέσιμος ένας πόρος.

semaphore_up: αυξάνει την τιμή ενός σημαφόρου. Χρησιμοποιείται για να σηματοδοτήσει ότι ένας πόρος είναι πλέον διαθέσιμος.

β) Αρχείο passenger.c

Υλοποιεί τη λογική για κάθε επιβάτη, ο οποίος: περιμένει την σειρά του στην ουρά, αποφασίζει αν θα αλλάξει γνώμη, επιβιβάζεται σε λέμβο με την προϋπόθεση ότι είναι η σειρά του.

Αναμονή στην ουρά: Ο επιβάτης περιμένει την ενεργοποίηση του passenger_queue (semaphore_down(passenger_queue);)

Αλλαγή γνώμης: ο επιβάτης μπορεί τυχαία να αλλάξει γνώμη και να επιστρέψει στο τέλος της ουράς. `(semaphore_up(passenger_queue);)` Χρησιμοποιήθηκε η συνάρτηση `rand`, ώστε να επιστρέφεται τυχαίος ακέραιος αριθμός.

`Rand() % 20 == 0` → το % είναι ο τελεστής modulus. Το `rand() % 20` επιστρέφει το υπόλοιπο της διαίρεσης του αποτελέσματος του `rand()` με το 20. Αυτό σημαίνει ότι το αποτέλεσμα θα είναι ένας αριθμός από 0 έως 19. Η συνθήκη ελέγχει αν το αποτέλεσμα της πράξης `rand() % 20` είναι ίσο με 0. Αυτό συμβαίνει με πιθανότητα 1 στις 20 περιπτώσεις, δηλαδή 5%.

Επιβίβαση αν δεν αλλάξει γνώμη: μειώνεται η διαθέσιμη χωρητικότητα και ο επιβάτης επιβιβάζεται `(max_seats--; if (max_seats == 0) { semaphore_up(boat_queue);})`

γ) Αρχείο `launch.c`

Αποτελεί το κύριο πρόγραμμα και διαχειρίζεται : τη δημιουργία και εκκίνηση των νημάτων για επιβάτες και λέμβους, την αρχικοποίηση των σημαφόρων και τον τερματισμό του προγράμματος όταν όλοι οι επιβάτες έχουν επιβιβαστεί.

Αρχικοποίηση Σημαφόρων: Οι σημαφόροι `passenger_queue`, `boat_queue`, και `mutex` αρχικοποιούνται μέσω της `create_semaphore`

Δημιουργία Νημάτων: Δημιουργούνται νήματα για τις λέμβους και τους επιβάτες.

Αναμονή για ολοκλήρωση νημάτων: Χρησιμοποιείται η `pthread_join` για να περιμένει το κύριο πρόγραμμα την ολοκλήρωση όλων των νημάτων επιβατών. `(for (int i = 0; i < total_passengers; i++) {pthread_join(passengers[i], NULL);})`

Καταστροφή Σημαφόρων : Όλοι οι σημαφόροι καταστρέφονται στο τέλος για να απελευθερωθούν οι πόροι.

Η σωστή επικοινωνία των αρχείων επιτυγχάνεται ως εξής:

α) Όπως αναφέραμε το αρχείο `ios_utils.h` λειτουργεί ως header file. Λειτουργεί δηλαδή ως γέφυρα μεταξύ των αρχείων κώδικα. Σε αυτό περιλαμβάνονται οι δηλώσεις συναρτήσεων που πρέπει να είναι προσβάσιμες από πολλαπλά αρχεία.

β) Κοινές μεταβλητές, όπως οι σημαφόροι (`passenger_queue`, `boat_queue`, `mutex`) χρησιμοποιούνται για συγχρονισμό, δηλώνονται ως `extern` στα αρχεία που τις χρειάζονται. Αυτό σημαίνει ότι η πραγματική τους αρχικοποίηση γίνεται σε ένα αρχείο, αλλά είναι προσβάσιμες από άλλα αρχεία. Στο αρχείο `launch.c` αρχικοποιούνται οι σημαφόροι και στο `passenger.c`, οι ίδιες μεταβλητές δηλώνονται ως `extern`.

γ) Τα νήματα δημιουργούνται στο `launch.c`, αλλά η λογική τους βρίσκεται στο `passenger.c` και στο `boat_thread` (μέρος του `launch.c`).

δ) Υπάρχει σωστή διαχείριση των κρίσιμων περιοχών, για να μην οδηγούμαστε σε αδιέξοδα. Οι κρίσιμες περιοχές προστατεύονται με τη χρήση του σημαφόρου `mutex`. Αυτό διασφαλίζει ότι μόνο ένα νήμα μπορεί να τροποποιήσει τη μεταβλητή τη φορά, αποφεύγοντας έτσι τα `race conditions`.

Η εντολή μεταγλώττισης που χρησιμοποιήθηκε για να συνδυαστούν όλα τα αρχεία και να δημιουργηθεί το εκτελέσιμο πρόγραμμα είναι η εξής: `gcc -o launch launch.c passenger.c -lpthread`

Τα αρχεία κεφαλίδας (`ipc_util.h`), περιλαμβάνονται κατά τη μεταγλώττιση με τη χρήση του `#include` μέσα στα αρχεία `.c`

Ακολουθεί παράδειγμα εκτέλεσης του προγράμματος.

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents$ cd semaphores_workspace3
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/semaphores_workspace3$ ./launch
Enter total passengers: 100
Enter number of boats: 8
Enter boat capacity: 10
Passenger 1 is boarding.
Passenger 2 is boarding.
Passenger 3 is boarding.
Passenger 4 is boarding.
Passenger 5 is boarding.
Passenger 6 is boarding.
Passenger 7 is boarding.
Passenger 8 is boarding.
Passenger 9 is boarding.
Passenger 10 is boarding.
Boat is departing.
Boat is returning.
Passenger 11 is boarding.
Passenger 12 is boarding.
Passenger 13 changed their mind and is moving to the back of the queue.
Passenger 14 is boarding.
Passenger 15 is boarding.
Passenger 16 is boarding.
Passenger 17 is boarding.
Passenger 18 is boarding.
Passenger 19 is boarding.
Passenger 20 is boarding.
Passenger 22 is boarding.
Boat is departing.
Passenger 23 is boarding.
Boat is returning.
Passenger 21 is boarding.
Passenger 24 is boarding.
Passenger 25 is boarding.
Passenger 26 is boarding.
Passenger 27 is boarding.
Passenger 28 is boarding.
Passenger 29 changed their mind and is moving to the back of the queue.
Passenger 30 is boarding.
Passenger 31 is boarding.
Passenger 32 is boarding.
Passenger 33 is boarding.
Boat is departing.
Boat is returning.
Passenger 34 is boarding.
Passenger 35 is boarding.
Passenger 36 is boarding.
Passenger 37 is boarding.
Passenger 38 is boarding.
```

Passenger 39 is boarding.
Passenger 40 is boarding.
Passenger 41 is boarding.
Passenger 42 is boarding.
Passenger 43 is boarding.
Boat is departing.
Boat is returning.
Passenger 44 is boarding.
Passenger 45 is boarding.
Passenger 46 is boarding.
Passenger 47 is boarding.
Passenger 48 is boarding.
Passenger 49 is boarding.
Passenger 50 is boarding.
Passenger 51 is boarding.
Passenger 52 is boarding.
Passenger 53 is boarding.
Boat is departing.
Boat is returning.
Passenger 54 is boarding.
Passenger 55 is boarding.
Passenger 56 is boarding.
Passenger 57 is boarding.
Passenger 58 is boarding.
Passenger 59 is boarding.
Passenger 60 is boarding.
Passenger 61 is boarding.
Passenger 62 is boarding.
Passenger 63 is boarding.
Boat is departing.
Boat is returning.
Passenger 64 is boarding.
Passenger 65 is boarding.
Passenger 66 is boarding.
Passenger 67 is boarding.
Passenger 68 is boarding.
Passenger 69 is boarding.
Passenger 70 is boarding.
Passenger 71 is boarding.
Passenger 72 is boarding.
Passenger 73 is boarding.
Boat is departing.
Boat is returning.
Passenger 74 is boarding.
Passenger 75 is boarding.
Passenger 76 is boarding.
Passenger 77 is boarding.
Passenger 78 is boarding.
Passenger 79 is boarding.
Passenger 80 is boarding.

```
Passenger 12 is boarding.  
Passenger 80 is boarding.  
Passenger 81 is boarding.  
Passenger 82 is boarding.  
Passenger 83 is boarding.  
Boat is departing.  
Boat is returning.  
Passenger 84 is boarding.  
Passenger 85 is boarding.  
Passenger 86 is boarding.  
Passenger 87 is boarding.  
Passenger 88 is boarding.  
Passenger 89 changed their mind and is moving to the back of the queue.  
Passenger 90 is boarding.  
Passenger 91 is boarding.  
Passenger 92 is boarding.  
Passenger 94 is boarding.  
Passenger 93 is boarding.  
Boat is departing.  
Passenger 95 is boarding.  
Passenger 96 is boarding.  
Boat is returning.  
Passenger 97 is boarding.  
Passenger 98 is boarding.  
Passenger 99 is boarding.  
Passenger 100 is boarding.  
Passenger 13 is boarding.  
Passenger 29 is boarding.  
Passenger 89 is boarding.  
All passengers have been transported. Exiting.  
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/senaphores workspace3$ nano lau
```

Άσκηση 3 Χρονοπρογραμματισμός Διεργασιών και Διαχείριση Μνήμης

Ο κώδικας υλοποιεί τη διαχείριση διεργασιών με τον αλγόριθμο Round Robin και τη διαχείριση μνήμης μέσω μοντελοποίησης της μνήμης ως πίνακα μπλοκ των 1KB. Κάθε διεργασία απαιτεί συγκεκριμένο μέγεθος μνήμης και εκτελείται κυκλικά με βάση το quantum.

Προβλήματα και Επίλυση

1. **Αναπαράσταση Μνήμης:** Δημιουργήθηκε πίνακας MemoryBlock για τη διαχείριση της μνήμης.

```
#include <string.h>

#define TOTAL_MEMORY 512 // Total memory size in KB

typedef struct {
    int start;
    int size;
    bool free;
    int pid;
} MemoryBlock;

MemoryBlock memory[TOTAL_MEMORY];
```

2. **Απελευθέρωση Μνήμης:** Προστέθηκε συνάρτηση free_memory για την απελευθέρωση των μπλοκ μετά την ολοκλήρωση των διεργασιών.

```
55 void free_memory(int pid) {
56     for (int i = 0; i < TOTAL_MEMORY; i++) {
57         if (memory[i].pid == pid) {
58             memory[i].free = true;
59             memory[i].pid = -1;
60         }
61     }
62 }
63
```

3. **Deadlock:** Υλοποιήθηκε έλεγχος για deadlock όταν καμία διεργασία δεν μπορεί να εκτελεστεί.
4. **Οπτικοποίηση:** Προστέθηκε η display_memory για την εμφάνιση της κατάστασης της μνήμης.

```

63
64 void display_memory() {
65     printf("Memory: [");
66     for (int i = 0; i < TOTAL_MEMORY; i++) {
67         if (!memory[i].free) {
68             printf("%d", memory[i].pid);
69         } else {
70             printf("_");
71         }
72     }
73     printf("]\n");
74 }

```

Σχεδιασμός

- **Αρχικοποίηση:** Δημιουργία πίνακα μνήμης και εισαγωγή διεργασιών.
- **Εκτέλεση:** Έλεγχος μνήμης, εκτέλεση διεργασιών με βάση το quantum, και εναλλαγή με τη σειρά.
- **Απελευθέρωση:** Απελευθέρωση μνήμης όταν ολοκληρωθεί μια διεργασία.
- **Αποτελέσματα:** Υπολογισμός και εμφάνιση χρόνων (turnaround, waiting) και κατάστασης μνήμης.

Συμπέρασμα

Η υλοποίηση διαχειρίζεται σχεδόν επιτυχώς την επίλυση της άσκησης. Δηλαδή είναι λειτουργικός μόνο εαν η συνολική μνήμη που χρειάζονται οι διεργασίες είναι μικρότερη του 512KB

Άσκηση 4

α) Αλγόριθμος FCFS

Διεργασίες	Χρόνος Αφίξης	Διάρκεια Έκτελεσης	Χρόνος Ολοκλήρωσης	Χρόνος Αναμονής	Χρόνος Απόκρισης
A	0	6	6	0	0
B	2	4	8	4	4
Γ	3	1	8	7	7
Δ	4	3	10	7	7
E	5	5	14	9	9
Z	6	7	20	13	13

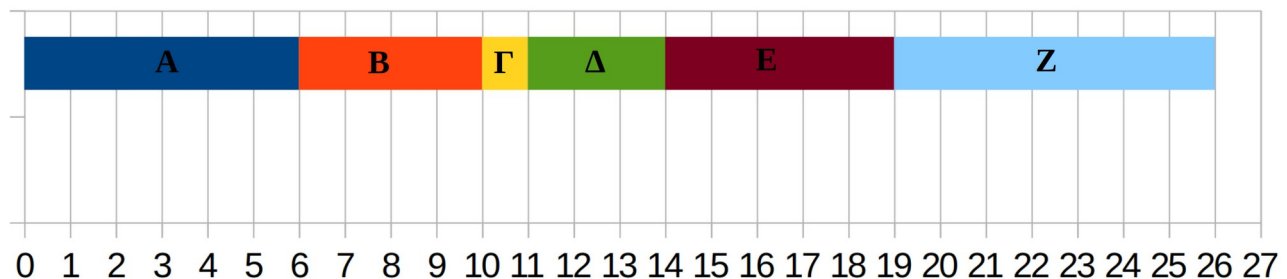
Μέσος χρόνος ολοκλήρωσης = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{αφίξης}})/n_{\text{διεργασίες}} = 11\text{ms}$

Μέσος χρόνος αναμονής = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{αφίξης}} - t_{\text{διάρκεια εκτέλεσης}})/n_{\text{διεργασίες}} = 6,6\text{ms}$

Μέσος χρόνος απόκρισης = $\Sigma(t_{\text{που άρχισε}} - t_{\text{αφίξης}})/n_{\text{διεργασίες}} = 6,6\text{ms}$

Θεματικές αλλαγές = αριθμός “κουτιών” χρόνου – 1 = 5

Διάγραμμα Gantt



β) Αλγόριθμος SJF

Διεργασίες	Χρόνος Αφίξης	Διάρκεια Έκτελεσης	Χρόνος Ολοκλήρωσης	Χρόνος Αναμονής	Χρόνος Απόκρισης
A	0	6	6	0	0
B	2	4	12	8	8
Γ	3	1	4	3	3
Δ	4	3	6	3	3
E	5	5	14	9	9
Z	6	7	20	13	13

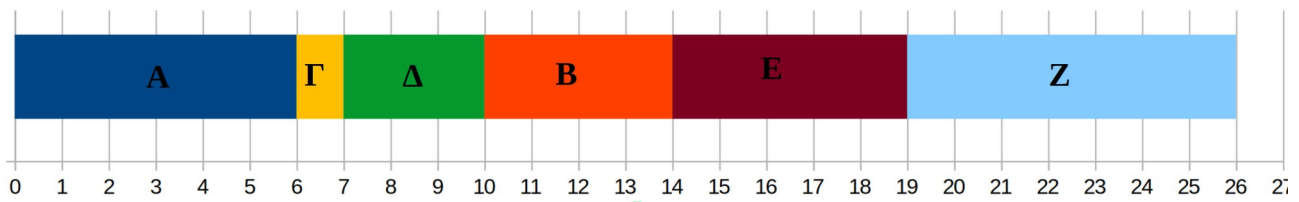
Μέσος χρόνος ολοκλήρωσης = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{αφίξης}})/n_{\text{διεργασίες}} = 10,3\text{ms}$

Μέσος χρόνος αναμονής = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{αφίξης}} - t_{\text{διάρκεια εκτέλεσης}})/n_{\text{διεργασίες}} = 6\text{ms}$

Μέσος χρόνος απόκρισης = $\Sigma(t_{\text{που άρχισε}} - t_{\text{αφίξης}})/n_{\text{διεργασίες}} = 6\text{ms}$

Θεματικές αλλαγές = αριθμός “κουτιών” χρόνου – 1 = 5

Διάγραμμα Gantt



γ) SRTF

Διεργασίες	Χρόνος Αφίξης	Διάρκεια Έκτελεσης	Χρόνος Ολοκλήρωσης	Χρόνος Αναμονής	Χρόνος Απόκρισης
A	0	6	7	1	0
B	2	4	12	8	8
Γ	3	1	1	0	0
Δ	4	3	6	3	3
E	5	5	14	9	9
Z	6	7	20	13	13

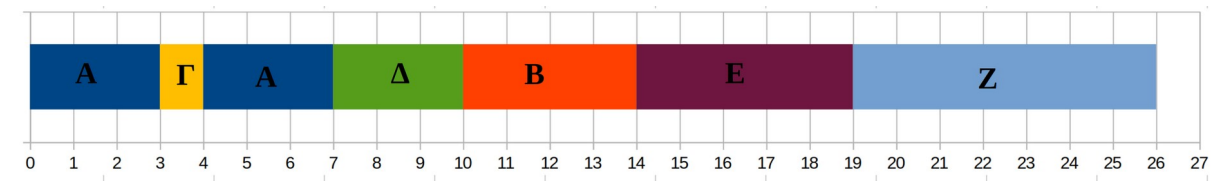
Μέσος χρόνος ολοκλήρωσης = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{αφίξης}})/n_{\text{διεργασίες}} = 10\text{ms}$

Μέσος χρόνος αναμονής = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{αφίξης}} - t_{\text{διάρκεια εκτέλεσης}})/n_{\text{διεργασίες}} = 5,6\text{ms}$

Μέσος χρόνος απόκρισης = $\Sigma(t_{\text{που άρχισε}} - t_{\text{αφίξης}})/n_{\text{διεργασίες}} = 5,5\text{ms}$

Θεματικές αλλαγές = αριθμός “κουτιών” χρόνου – 1 = 6

Διάγραμμα Gantt



γ) Round Robin

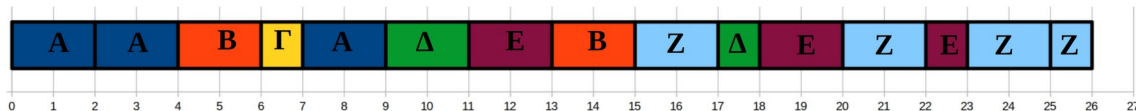
Διεργασίες	Χρόνος Αφίξης	Διάρκεια Έκτελεσης	Χρόνος Ολοκλήρωσης	Χρόνος Αναμονής	Χρόνος Απόκρισης
A	0	6	9	3	0
B	2	4	13	9	2
Γ	3	1	4	3	3
Δ	4	3	14	11	5
E	5	5	18	13	6
Z	6	7	20	13	9

Μέσος χρόνος ολοκλήρωσης = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{άφιξης}})/n_{\text{διεργασίες}} = 13\text{ms}$

Μέσος χρόνος αναμονής = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{άφιξης}} - t_{\text{διάρκεια εκτέλεσης}})/n_{\text{διεργασίες}} = 8,6\text{ms}$

Μέσος χρόνος απόκρισης = $\Sigma(t_{\text{που άρχισε}} - t_{\text{άφιξης}})/n_{\text{διεργασίες}} = 4\text{ms}$

Θεματικές αλλαγές = αριθμός “κουτιών” χρόνου – 1 = 14



Πιο αναλυτικά βλέπουμε την κατάσταση στον παρακάτω πίνακα, θεωρούμε πώς αν μια χρονική στιγμή 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.

Χρονικές στιγμές	Διεργασία στην ΚΜΕ	Διεργασία που έρχεται	Ουρά
t=0	-	A	A
t=0-1	A	-	-
t=1-2	A	-	-
t=2-3	A	B	B
t=3-4	A	Γ	B,Γ
t=4-5	B	Δ	Γ,A,Δ
t=5-6	B	E	Γ,A,Δ,E
t=6-7	Γ	Z	A,Δ,E,B,Z
t=7-8	A	-	Δ,E,B,Z
t=8-9	A	-	Δ,E,B,Z
t=9-10	Δ	-	E,B,Z
t=10-11	Δ	-	E,B,Z
t=11-12	E	-	B,Z,Δ
t=12-13	E	-	B,Z,Δ
t=13-14	B	-	Z,Δ,E
t=14-15	B	-	Z,Δ,E
t=15-16	Z	-	Δ,E
t=16-17	Z	-	Δ,E
t=17-18	Δ	-	E,Z
t=18-19	E	-	Z
t=19-20	E	-	Z
t=20-21	Z	-	E
t=21-22	Z	-	E
t=22-23	E	-	Z
t=23-24	Z	-	-
t=24-25	Z	-	-
t=25-26	Z	-	-

δ) LRTEP

Διεργασίες	PID	Χρόνος Αφίξης	Διάρκεια Έκτελεσης	Χρόνος Ολοκλήρωσης	Χρόνος Αναμονής	Χρόνος Απόκρισης
A	3	0	6	24	18	0
B	1	2	4	19	15	0
Γ	2	3	1	20	19	19
Δ	5	4	3	22	19	10
E	4	5	5	20	15	0
Z	1	6	7	16	9	0

Χρονικές στιγμές	Διεργασία στην ΚΜΕ	Ουρά[υπολ. διαρκ.]	Αρχική Διάρκεια	Υπολειπόμενη διάρκεια
t=0	-	A[6]	-	-
t=0-1	A	-	6	5
t=1-2	A	B[4]	5	4
t=2-3	B	A[4], Γ[1]	4	3
t=3-4	A	B[3], Δ[3], Γ[1]	4	3
t=4-5	B	E[5], A[3], Δ[3], Γ[1]	3	2
t=5-6	E	Z[7], A[3], Δ[3], B[2], Γ[1]	5	4
t=6-7	Z	E[4], A[3], Δ[3], B[2], Γ[1]	7	6
t=7-8	Z	E[4], A[3], Δ[3], B[2], Γ[1]	6	5
t=8-9	Z	E[4], A[3], Δ[3], B[2], Γ[1]	5	4
t=9-10	Z	E[4], A[3], Δ[3], B[2], Γ[1]	4	3
t=10-11	E	Z[3], A[3], Δ[3], B[2], Γ[1]	4	3
t=11-12	Z	E[3], A[3], Δ[3], B[2], Γ[1]	3	2
t=12-13	E	A[3], Δ[3], B[2], Z[2], Γ[1]	3	2
t=13-14	A	Δ[3], B[2], Z[2], E[2], Γ[1]	3	2
t=14-15	Δ	B[2], Z[2], A[2], E[2], Γ[1]	3	2
t=15-16	B	Z[2], A[2], E[2], Δ[2], Γ[1]	2	1
t=16-17	Z	A[2], E[2], Δ[2], B[1], Γ[1]	2	1
t=17-18	A	E[2], Δ[2], B[1], Z[1], Γ[1]	2	1
t=18-19	E	Δ[2], B[1], Z[1], Γ[1], A[1]	2	1
t=19-20	Δ	B[1], Z[1], Γ[1], A[1], E[1]	2	1
t=20-21	B	Z[1], Γ[1], A[1], E[1], Δ[1]	1	0
t=21-22	Z	Γ[1], A[1], E[1], Δ[1]	1	0
t=22-23	Γ	A[1], E[1], Δ[1]	1	0
t=23-24	A	E[1], Δ[1]	1	0

t=24-25	E	$\Delta[1]$	1	0
t=25-26	Δ	-	0	0

Μέσος χρόνος ολοκλήρωσης = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{άφιξης}})/n_{\text{διεργασίες}} = 20,16\text{ms}$

Μέσος χρόνος αναμονής = $\Sigma(t_{\text{που τελείωσε}} - t_{\text{άφιξης}} - t_{\text{διάρκεια εκτέλεσης}})/n_{\text{διεργασίες}} = 15,83\text{ms}$

Μέσος χρόνος απόκρισης = $\Sigma(t_{\text{που άρχισε}} - t_{\text{άφιξης}})/n_{\text{διεργασίες}} = 4,8\text{ms}$

Θεματικές αλλαγές = αριθμός “κουτιών” χρόνου – 1 = 25

