Λειτουργικά Συστήματα – Τμήμα Περιττών 2018

Εργασία 3

ΜΑΡΑΓΚΟΖΑΚΗΣ ΓΕΩΡΓΙΟΣ – 1115201500089

Δομή Εργασίας:

myport.c, port_master.c, vessel.c, monitor.c, make_vessels.c, structures.c, structures.h

structures.h:

typedef struct **Vessel_info**: Αναπαρίσταται το κάθε πλοίο

typedef struct **Mooring_Area**: Αναπαρίσταται η κάθε θέση για ένα πλοίο, κάθε Mooring_Area, έχει ένα χαρακτηριστικό type για τους τύπους πλοίων που μπορεί να εξυπηρετήσει.

typedef struct **Public Ledger**: Αναπαρίστανται οι διαθέσιμες θέσεις του λιμανιού για τα πλοία. Υπάρχουν 3 τύποι θέσεων, Small, Medium, Large, τα οποία είναι δείκτες στις διαθέσιμες θέσεις. Επιπλέον, υπάρχει ένας πίνακας int στον οποίο αποθηκεύεται ο αριθμός των μέγιστων διαθέσιμων θέσεων για κάθε τύπο πλοίου του λιμανιού.

typedef struct **Waiting_Vessel**: Βοηθητική δομή στην οποία κρατούνται τα στοιχεία διαφορετικών τύπων πλοίων που επιθυμούν να εισέλθουν στο λιμάνι, καθώς και οι προτεραιότητες τους.

typedef struct **SHARED_MEMORY**: Το τμήμα της κοινής μνήμης στο οποίο αποθηκεύονται οι απαιτούμενες δομές καθώς και οι semaphores.

Δομές:

Vessel_info vessel_in_S, vessel_in_M, vessel_in_L : κρατούνται οι πληροφορίες των πλοίων που περιμένουν σε τρεις “ουρές”, ανάλογα τον τύπο των πλοίων.

Vessel_info vessel_out : κρατούνται οι πληπροφορίες του πρώτου πλοίο στην “ουρά” των πλοίων που επιθυμούν να εξέλθουν του λιμανιού.

Double vessel_sum_cost : Το συνολικό εισόδημα του λιμανιού

Public_Ledger public_ledger : Η **public ledger** η οποία ενημερώνεται με τα πλοία που βρίσκονται στο λιμάνι την τωρινή κατάσταση.

Semaphores:

sem_t read_write;	χρησιμοποιείται για διάβασμα-γράψιμο και εννίοεται για κλείδωμα
sem_t vessel_sem_in;	ειδοποιεί τον port_master για την άφιξη ενός vessel που επιθυμεί να εισέλθει στο λιμάνι
sem_t vessel_sem_out;	ειδοποιεί τον port_master για την επιθυμία αναχώρησης ενός vessel
sem_t vessel_sem_L;	Δεσμέυεται από το πρώτο **Large** που επιθυμεί να εισέλθει στο λιμάνι και μπλοκάρει τα επόμενα έως ότου αυτό ελλιμενιστεί
sem_t vessel_sem_M;	Δεσμέυεται από το πρώτο **Medium** που επιθυμεί να εισέλθει στο λιμάνι και μπλοκάρει τα επόμενα έως ότου αυτό ελλιμενιστεί
sem_t vessel_sem_S;	Δεσμέυεται από το πρώτο **Small** που επιθυμεί να εισέλθει στο λιμάνι και μπλοκάρει τα επόμενα έως ότου αυτό ελλιμενιστεί
sem_t portin_sem_L;	Μπλοκάρεται ένα **Large** μέχρις ότου του δοθεί άδεια για είσοδο στο λιμάνι
sem_t portin_sem_M;	Μπλοκάρεται ένα **Medium** μέχρις ότου του δοθεί άδεια για είσοδο στο λιμάνι
sem_t portin_sem_S;	Μπλοκάρεται ένα **Small** μέχρις ότου του δοθεί άδεια για είσοδο στο λιμάνι
sem_t port_sem_out;	Μπλοκάρεται μέχρις ότου δεν υπάρχει πλοίο που να βρίσκεται σε διαδικασία κίνησης – mantime εντός του λιμανιού

sem_t mutex1; Μπλοκάρει τα πλοία που επιθυμούν να εξέλθουν του λιμανιού έως ότου ο port_master εκτελέσει τις απαραίτητες διεργασίες για την έξοδο του πρώτου στην ουρά.

sem_t mutex2; Χρησιμποιείται για συγχρονισμό μεταξύ του **port_master** και του **vessel**, κατά τη διαδικασία εισαγωγής ενός **vessel** στο Mooring_Area.

sem_t mutex3; "Κλειδί" το οποίο ενεργοποιεί ο **port_master** και δίνεται πρόσβαση στον **vessel** για ασφαλή αποχώρηση από το λιμάνι, αφού έχει γίνει έλεγχος για ελεύθερη δίοδο.

sem_t monitor; Ενεργοποιείται/Μπλοκάρεται ώστε να τεθεί σε λειτουργία/τερματιστεί ο **monitor.c**

myport.c: Δημιουργείται το τμήμα της κοινής μνήμης και οι κατάλληλοι semaphores μέσα σε αυτή, καθώς και αρχικοποιούνται οι μεταβλητές της. Επιπλέον, μέσω της **write_id_to_file** γράφεται σε ένα αρχείο το id της κοινής μνήμης ώστε να το διαβάσουν οι υπόλοιπες διεργασίες και να αποκτήσουν πρόσβαση σε αυτή. Τέλος, μέσω της **make_new_process** δημιουργείται μια νέα διεργασία η οποία στη συνέχεια επικαλύπτεται με το εκτελέσιμο του **port_master**.

port_master.c: Αρχικά, δημιουργείται ένα αρχείο public_ledger.txt στο οποίο κρατείται το ιστορικό της public_ledger του λιμανιού. Έπειτα καλείται η **port_master_server** η οποία αναλαμβάνει το κύριο όγκο δουλειάς του port_master, καθώς μέσω αυτής διαχειρίζονται τα αιτήματα των πλοίων, για είσοδο ή έξοδο από το λιμάνι, καθώς και οι επιμέρους λειτουργίες που απαιτούν τα παραπάνω αιτήματα. Κατά την εκτέλεση δεν χρησιμοποιείται κάποιο αρχείο charges μιας και τα απαραίτητα στοιχεία βρίσκονται στο configuration_file.txt

make_vessels.c: Μέσα από εδώ δημιουργούνται "ορδές" πλοίων που προσεγγίζουν το λιμάνι, επιλέγοντας για κάθε πλοίο ο χρήστης, το type και το postype του. Αναλυτικότερα, δημιοργείται ένας πίνακας ανάλογα τον αριθμό των πλοίων που θέλει να βάλει ο χρήστης και μέσω της **give_vessel_info** δινονται στα πλοία τα κατάλληλα χαρακτηριστικά. Στη συνέχεια μέσω της **make_processes** δημιουργείται αντίστοιχος αριθμός διεργασιών οι οποίες επικαλύπτονται με το εκτελέσιμο του **vessel.c** παίρνοντας η κάθε μια τα στοιχεία ενός πλοίου.

Vessel.c: Αρχικά αυξάνεται ο counter waiting_vessels και στη συνέχεια μπλοκάρεται από το vessel_sem_… ανάλογα τον τύπο του μέχρι να είναι πρώτο στην "ουρά" της κατηγορίας του. Έπειτα, μόλις είναι στην πρώτη θέση της ουράς της κατηγορίας του γράφει τα στοιχεία του στο αντίστοιχο κομμάτι της shared memory για να έχει πρόσβαση σε αυτά ο port_master. Μετά από αυτό μπλοκάρεται από τον portin_sem_… μέχρις ότου του δωθεί άδεια για ελιμενισμό. Επιπλέον όταν του δοθεί άδεια και αφού εκτελεστούν οι απαραίτητες εντολές για mantime και parkperiod, τότε στα μισά του parkperiod, ζητήται το κόστος που διαμορφώθηκε μέχρι αυτή τη στιγμή. Τέλος όταν θελήσει να αποχωρήσει στέλνει σήμα στον port_master, μέσω του vessel_sem_out, και στη συνέχεια περιμένει έως ότου λάβει εντολή αποχώρησης από το λιμάνι. Φεύγοντας από το λιμάνι ενημερώνεται για το τελικό κόστος.

Monitor.c: Για την κλήση της print_times print_statimes ακολουθείται η εξής λογική. Γίνεται

```
sleep(times)
print_times
count++
if (statimes – (count*times) < times)
        sleep(st)
        print_statimes
        count = 0
```

Structures.c:

read_waiting_vessels: Διαβάζει για κάθε τύπο πλοίου το πρώτο από την ουρά αυτών και τα τοποθετεί σε ένα πίνακα.

sorting_the_array, swap: Ταξινόμηση των στοιχείων του πίνακα που διαβάστηκαν παραπάνω κατά αύξουσα σειρά βάσει του priority, κάθε πλοίο χαρακτηριζέται από ένα μοναδικό αύξων priority, το οποίο είναι το id της κάθε διεργασίας. Για αυτές τις δύο συναρτήσεις χρησιμοποιήθηκε ο αλγόριθμος bublesort, από την πηγή: GeeksforGeeks.org

find_empty_seat: Βρίσκεται και επιστρέφεται η θέση του πίνακα που δεν υπάρχει πλοίο τη συγκεκριμένη περίοδο.

store_vessel_to_seat: Ανάλογα τον τύπο του πλοίου ξεπμπλοκάρεται από την αναμονή το vessel, καλείται για εξυπηρέτηση από τον **port_master,** επιπλέον δίνεται πρόσβαση στο επόμενο πλοίο του αντίστοιχου type να γράψει τα στοιχεία του στη στην πρώτη θέση της λίστας αναμονής. Έπειτα, με βάση τον τύπο του πλοίου βρίσκεται ελεύθερη θέση για αυτό και τοποθετείται στην public_ledger μαζί με στοιχεία όπως το όπως το arrival_time, και το service_time.

check_empty_seat: Ελέγχεται αν υπάρχει διαθέσιμη θέση για το πλοίο με βάση το type του, αλλιώς αν δεν βρεθεί ελέγχεται για το διαφορετικό postype του.

port_master_server: Ελέγχεται αν υπάρχει αίτημα από πλοία για είσοδο ή έξοδο από το λιμάνι. Δίνεται προτεραιότητα στην έξοδο των πλοίων. Αν δεν υπάρχει κανένα αίτημα, ερωτάται ο χρήστης αν θέλει να περιμένει για ενδεχόμενη άφιξη πλοίων αλλίως τερματίζει.
1) Αν υπάρχουν πλοία για είσοδο στο λιμάνι, τότε με την read_waiting_vessels διαβάζονται τα 3 πρώτα διαφορετικού τύπου που περιμένουν και σορτάρονται ώστε πρώτο στον πίνακα να είναι αυτό με το μικρότερο priority. Στη συνέχεια ελέγχεται αν υπάρχει ελεύθερη θέση για το πρώτο, αλλιώς γίνεται έλεγχος για το δεύτερο. **Ακολουθείται η πολιτική FCFS, αλλά όταν δεν υπάρχει διαθέσιμη θέση για το πρώτο, ακολουθείται η πολιτική του Availability.** Τέλος, αφού βρεθεί η θέση τότε γίνεται έλεγχος για ελεύθερη δίοδο προς το εσωτερικό του λιμανιού μέσω του **sem_port_out** και μετά γράφεται το πλοίο στο λιμάνι. Αν δεν υπάρχει καμία διαθέσιμη θέση για τα πλοία στην ουρά τότε γίνεται ένα sleep(3), ώστε στο χρονικό αυτό διάστημα να έχει πιθανώς εξέλθει του λιμανιού κάποιο πλοίο.
2) Αν υπάρχουν πλοία για έξοδο τότε βρίσκεται η θέση στην οποία είχαν ελλιμενιστεί, υπολογίζεται το τελικό κόστος και γίνονται οι κατάλληλες αλλαγές στο public_ledger και τέλος δίνεται πρόσβαση στο επόμενο πλοίο που περιμένει για έξοδο να προχωρήσει στη διαδικασία. Εδώ, γράφεται σε ένα αρχείο το ιστορικό του public_ledger.

find_vessel_with_ID: Ψάχνει και βρίσκει τη θέση του vessel με βάση το ID του μέσα στους πίνακες του Mooring_Area , του public_ledger.

print_current_arays: Τυπώνεται η τωρινή κατάσταση του public_ledger

enable_disable_monitor: Ενεργοποιείται/Απενεργοποιείται ο monitor_sem για να εκτελεστεί/τερματιστεί ο monitor.

calc_average_waiting_time: Υπολογίζεται ο μέσος όρος αναμονής για κάθε τύπο πλοίου και για όλα τα πλοία μαζί.

calc_vessels_cost: Υπολογίζεται το κόστος που θα κληθεί να πληρώσει το πλοίο για το parkperiod που έκατσε στο λιμάνι. **Η πολιτική χρέωσης είναι ότι μέχρι 30 λεπτά χρεώνεται με το Κ ποσό, ανάλογα τη θέση. Στη συνέχεια για όσο χρόνο κάτσει το κόστος είναι:**

Σ.Κ. = (parkperiod/30)*κ + κ, αν το υπάρχει υπόλοιπο στη διαίρεση,
αλλιώς Σ.Κ. = (parkperiod/30)*κ.

Παρατηρήσεις και Εκτέλεση:
Χρησιμοποιείται Makefile
Κατά την εκτέλεση δεν παρατηρήθηκαν error leaks
Εντολές Εκτέλεσης με την παρακάτω σειρά, σε 3 διαφορετικά τηλέτυπα:
./myport -l configuration.txt
./monitor -d times -t statimes -s shared_memory_id.csv
./make_vessels shared_memory_id.csv n, όπου n ο αριθμός των πλοίων-vessels που θέλουμε να δημιουργήσουμε
Στη συνέχεια στο myport δίνουμε W ώστε να λάβει τα μηνύματα από τα vessels, ή E για να τερματίσει το πρόγραμμα.