ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΡΟΦΟΡΙΚΗΣ
ΤΟΜΕΑΣ ΛΟΓΙΚΟΥ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα Ι

Αναφορά Project

Σπύρος Καφτάνης (5542)

1. Υλοποίηση με βάση διαδικασίες

Στην υλοποίηση αυτή υπάρχει ένας server και κάποιοι clients οι οποίοι συνδέονται σε αυτόν μέσω sockets. Ο server καλείται ως:

./gameserver -p <num_of_players> -i <game_inventory> -q <quota_per_player>

Ο χρήστης πρέπει να δώσει τα στοιχεία που θέλει με τη συγκεκριμένη μορφή διαφορετικά ο server τερματίζει με μήνυμα λάθους. Αφού ελεγθεί η εγκυρότητα των ορισμάτων ο server περνάει στη δεύτερη φάση.

Εκεί, ο server ανοίγει το αρχείο που έχει ορίσει ο χρήστης και έπειτα υποθέτοντας πως το αρχείο είναι στο επιτρεπόμενο format, αποθηκεύει τις πληροφορίες που έρχονται από αυτό σε πίνακες. Με τη συνάρτηση write2SM, τα στοιχεία των πινάκων αυτών γράφονται σε μια shared memory με κλειδί 5679, μαζί με το quota και με το πλήθος των online παιχτών αυτή τη στιγμή, έτσι ώστε οι clients να μπορούν να έχουν πρόσβαση στα δεδομένα αυτά. Έπειτα όταν έρθει κάποιο αίτημα από τον client, δημιουργείται μια νέα forked process.

Όταν βρισκόμαστε σε child process (childpid=0), ο πρώτος σηματοφόρος, ο οποίος χρησιμοποιείται σαν mutex κλειδώνει, μαζί με τον δεύτερο, με σκοπό την αποφυγή του αμοιβαίου αποκλεισμού.

Όταν συνδεθεί ο πρώτος χρήστης στον server, ο δεύτερος του ετοιμάζει την shared memory με όλα τα στοιχεία του inventory, το quota κτλ. Ο server ξεκλειδώνει τον semaphore 1 έτσι ώστε ο client να αναλάβει δράση!

Αφού πρώτα κλειδώσε εκ νέου τον semaphore-mutex ανοίξει μια σύνδεση με την κοινή μνήμη (5679), και χρησιμοποιώντας έναν αλγόριθμο που υλοποιήθηκε, αποθηκεύει τα στοιχεία σε πίνακες. Η shared memory μπορεί να διαβαστεί γράμμα γράμμα με έναν δείκτη *s και ο αλγόριθμος γνωρίζοντας το format του inventory, δημιουργεί έναν πίνακα με τα ονόματα των resources και έναν παράλληλο με τις αντίστοιχες ποσότες, όπως ακριβώς και οι πίνακες που έχουν τα αρχικά δεδομένα που διαβάστηκαν από τα αρχικά inventories στα αρχεία.

Ο client τώρα, εφόσον έχει κλειδώσει τον server, κάνει τους κατάλληλους ελέγχους των στοιχείων του server με των δικών του. Αν δε βρεθεί κάποιο πρόβλημα (λόγω quota, λόγω εσφαλμένου resources name κτλ), τότε ο client συνεχίζει να λειτουργεί αλλιώς τερματίζεται. Αν τα αποθέματα δεν αρκούν ο client δεν τερματίζεται, αλλά αρκείται με όσα υπάρχουν.

Ο semaphore ξεκλειδώνει. Ο client στέλνει στον server το αίτημά του (μόνο εάν έχει περάσει τη διαδικασία του ελέγχου). Ο server την ίδια στιγμή λαμβάνει το μήνυμα το αποθηκεύει σε αντίστοιχους με πριν πίνακες και καταναλώνει από το gameInventory. Τέλος ενημερώνει τη shared memory με τις νέες τιμές και ο δεύτερος semaphore ξεκλειδώνει επιτρέποντας σε κάποια άλλη διεργασία να ακολουθήσει την ίδια διαδικασία.

Έπειτα ελέγχετε εάν έχει συμπληρωθεί ο αριθμός των παιχτών που θέλουμε για να ξεκινήσει η παρτίδα. Εάν δεν έχει σε μιά άλλη shared memory (με τη χρήστη της συνάρτησης writeMessage2SM2) γράφεται η τιμή '1'. Όσο ο client διαβάζει τη συγκεκριμένη τιμή εμφανίζει το μήνυμα "Waiting for players...".. Όταν γραφτεί το '0' σημαίνει ότι έχουμε φτάσει στους παίχτες που θέλουμε, οπότε δεν εμφανίζεται πλέον αυτό το μήνυμα ξεκινάει σε ατέρμον βρόγχους στον sever και στον client μία client – server επικοινωνία κατά την οποία ο client στέλνει μηνύματα στον server και αυτός τα εμφανίζει.

2. Υλοποίηση με βάση νήματα

Στην υλοποίηση αυτή ο κάθε client εξυπηρετείται από threads, αντί για από κάποια fork process. Το πρώτο thread που αναλαμβάνει μία νέα αίτηση είναι το countUsers, το οποίο αυξάνεται κατά ένα κάθε φορά που κάποιος νέος χρήστης συνδέεται στον server. Το "main thread" ονομάζεται process και εκτελεί όλες τις υπόλοιπες λειτουργίες.

Ψευδοκώδικας thread: process στον **server**

```
Αρχικοποίηση μεταβλητών
Αναμονή για τη σύνδεση όλων των παιχτών
Έναρξη παρτίδας όταν συμπληρωθούν οι  παίχτες
Διάβασμα αιτήματος από client
Αποκωδικοποίηση μηνύματος και αποθήκευσή του σε πίνακες
Κρίσιμη περιοχή {
      Έλεγχος διαθέσιμων πόρων και quota
      Κατανάλωση αν υπάρξει επιτυχημένος έλεγχος
}
Υποδοχή μηνυμάτων από τους clients
```

Μόνο ένα thread μπορεί να βρίσκεται στη κρίσιμη περιοχή του, Για να επιτευχθεί ο αμοιβαίος αποκλεισμός σε αυτή τη περίπτωση χρησιμοποιείται ένα απλό mutex που "κλειδώνει" όταν κάποιο thread μπαίνει στη περιοχή αυτή και "ξεκλειδώνει" όταν βγαίνει.

Πριν από τη δημιουργία του thread αυτού ο server, όπως ακριβώς έγινε και στην υλοποίηση με διαδικασίες, διαβάζει από το inventory που έχει δώσει ο χρήστης και αποθηκεύει τα στοιχεία του σε πίνακες και τα υπόλοιπα στοιχεία (όριο παιχτών, quota) σε μεταβλητές.

Ο **client** τώρα, εκτελεί επίσης τις ίδιες διαδικασίες που εκτελούσε και στην προηγούμενη υλοποίηση με τη διαφορά ότι αυτή τη φορά δεν επεξεργάζεται τίποτα. Το μόνο που κάνει αυτή τη φορά είναι να διαβάζει τα στοιχεία του inventory και να τα στέλνει στον server. Ο server αναλαμβάνει την επεξεργασία των δεδομένων.

Για τη κλήση του gameserver χρησιμοποιείται το αρχείο gameInventory και για τους players ένα από τα αρχεία playerInvernory1,2 ή οποίο άλλα αρχεία θέλει ο χρήστης τα οποία έχουν το κατάλληλο format.