Readme File

Εργασία 2 Προγραμματισμός Συστήματος 2013 – 2014

(Γλώσσα : C)

Περιεχόμενα :

---

1. Makefile - Εκτέλεση - Scripts

Το Makefile δημιουργεί τα εκτελέσιμα jobcommander και jobexecutorserver . Κάνει separate compilation και έχει clean mode .

Το εκτελέσιμο τρέχει είτε με ./jobcommander και μία από τις επιλογές που περιγράφονται στην εκφώνηση (issuejob <job>, κτλ) είτε μέσω του script ./multijob.sh και ένα η περισσότερα αρχεία .Τα αρχεία αυτά μέσα θα πρέπει να περιέχουν εντολές προς εισαγωγή στον Server.

Ο jobcommander κάνει fork exec το jobexecutorserver.

Το script allJobsStop.sh σταματά όλες τις εργασίες που τρέχουνε και αφαιρεί και όλες τις εργασίες από την λίστα αναμονής (waiting_list).

Τον κώδικα του while στο script multijob.sh τον βρήκα στο net ουσιαστικά.

---

2. Δομές

Ο jobexecutorserver έχει την εξής λίστα:

- Κάθε εργασία προς εκτέλεση έχει την εξής μορφή :

```
struct Job {
        int id ;
        char ** exec_matrix ;
        pid_t pid ;
        pid_t client_pid ;
```

```
};
```

Όπου id το αναγνωριστικό που αναθέτει ο Server στην κάθε εργασία,το exec_matrix πίνακας με τα ορίσματα της εργασίας,pid το αναγνωριστικό που αναθέτει το OS στην κάθε εργασία και client_pid το PID του jobcommander που έστειλε το αίτημα για αυτήν την εργασία.

- Ο κάθε κόμβος της λίστας είναι ως εξής :

```
struct Node {
        Job * job ;
        Node * next ;
};
```

- Τέλος η λίστα έχει μητρικό κόμβο ο οποίος έχει πληροφορία για το μέγεθος της λίστας και δείκτη στον πρώτο και τελευταίο κόμβο ώστε να προσθέτει κόμβο στο τέλος και να αφαιρεί από την αρχή(FIFO) σε O(1).

```
struct MotherNode {
        Node * first ;
        Node * last ;
        int size ;
};
```

3.Σύντομη περιγραφή κώδικα :

Ο jobcommander αρχικά δημιουργεί τον Server αν αυτός δεν υπάρχει ήδη .Αυτό γίνεται με έλεγχο ύπαρξης του checkfile.txt το οποίο δημιουργεί ο Server κατά την δημιουργία του .Ο Server κανει pause().Στη συνέχεια ο jobcommander περνάει από το πρώτο pipe (ClientToServerFifo) την εντολή που έχει πάρει απο τη γραμμή εντολών του .Μόλις γράψει ότι πρέπει στο pipe στέλνει σήμα στον Server(**SIGRTMIN**) .Ο Server τότε "ξυπνάει" και διαβάζει ότι χρειάζεται από το pipe (ClientToServerFifo) και στη συνέχεια δημιουργεί job το οποίο και προσθέτει στην ουρά αναμονής(waiting_list) αν πρόκειται για νέα εργασία (issuejob),η ακολουθεί διαφορετική συμπεριφορά αν πρόκειται για άλλη εντολή.Αν ο jobcommander στείλει μία εκ των poll η issuejob τότε με σημαφόρο (named semaphore) κολλάει στο δεύτερο pipe (ServerToClientFifo) και περιμένει να διαβάσει .Ο Server όταν γράψει στο pipe(ServerToClientFifo)κάνει up τον σημαφόρο και τότε ξεκολλάει ο jobcommander και διαβάζει από το pipe(ServerToClientFifo).Τέλος αν ο Server λάβει exit περιμένει αν υπάρχουνε running processes να τερματίσουν απελευθερώνει πόρους (λίστες) και κάνει exit.

Το αρχείο race.txt χρησιμοποιείται για να λύσει πρόβλημα συγχρονισμού του πρώτου jobcommander και του Server.

Ο Server μαζεύει τα παιδιά του με handling του σήματος **SIGCHLD** και αφαιρεί από την running_list τα σχετικά jobs.

4.Σχεδιαστικές επιλογές :

- 2 named pipes για επικοινωνία μεταξύ jobcommander και Server.
- signals για το pipe(ClientToServerFifo) με το σήμα **SIGRTMIN** .
- Named semaphore για το pipe(ServerToClientFifo) για πιο safe mode.