Λειτουργικά Συστήματα

2022 - 2023

2η Εργαστηριακή Άσκηση

Μέλη Ομάδας:

1) Όνομα : Αγγουρά Ρουμπίνη Μαρία
   AM: 1084634
   Email: up1084634@upnet.gr

2) Όνομα : Παυλόπουλος Ιάσονας
   AM: 1084565
   Email: up1084565@upnet.gr

Εισαγωγή

Πριν ξεκινήσουμε την παρουσίαση του Project, θα θέλαμε να σας ενημερώσουμε πως όλα τα resource files, όπως το αρχείο με τον κώδικα για το project 2 (scheduler.c), συμπεριλαμβανομένης και αυτής της αναφοράς μπορείτε να τα βρείτε στο παρακάτω Github Repository. Επίσης μπορείτε να δείτε τις αλλαγές που έχουν γίνει από την αρχή δημιουργίας του Project (commits).

Github Repository: https://github.com/Roumpini21/Project-2--Operating-Systems

Περιγραφή Κώδικα

Αρχικά πρωτού αρχίσουμε να υλοποιούμε κάποιο αλγόριθμο χρονοπρογραμματισμού, αρχικοποιήσαμε την δομή δεδομένων των διεργασιών μας (proc) με πεδία name, pid, priority, at (Arrival Time), bt (Burst Time), wt (Workload Time), state, status και δύο pointers next και prev οι οποίοι δείχνουν στην επόμενη και προηγούμενη διεργασία από αυτήν αντίστοιχα.

Έπειτα, αρχικοποιήσαμε την δομή της λίστας (queue) μας. Περιέχει δύο Pointers τύπου proc* με όνομα head και end που σηματοδοτούν την αρχή και το τέλος της ουράς μας αντίστοιχα.

Τέλος αρχικοποιήσαμε έναν proc* ώστε να αποθηκεύουμε τυχόν διεργασίες που θέλουμε να έχουμε πρόσβαση σε όλες τις συναρτήσεις μας.

Στην συνέχεια, στην main δημιουργήσαμε την βασική ουρά μας queue1 με την χρήση της συνάρτησης CreateQueue (στην οποία δεσμεύουμε χώρο για το struct μας), έναν File Pointer και έναν Integer option. Με βάση τα ορίσματα που δίνουμε στο terminal για την εκτέλεση του προγράμματος, είτε παίρνει το όνομα του αρχείου από το $3^{ο}$ ή το $4^{ο}$ όρισμα. Έπειτα συγκρίνοντας, με την χρήση της strcmp, το $2^{ο}$ όρισμα με το όνομα του αλγορίθμου που θέλουμε να χρησιμοποιήσουμε, επιλέγουμε τις επόμενες κινήσεις μας. Για παράδειγμα, αν επιλεχθεί ο αλγόριθμος Batch, εκτυπώνεται κατάλληλο μήνυμα και εκτελείται η fill_queue με ορίσματα την ουρά queue1, τον file pointer και το option (που θέτουμε ανάλογα την επιλογή του αλγορίθμου).

Η fill_queue είναι μία συνάρτηση η οποία ανάλογα την περίπτωση, γεμίζει την ουρά με processes, δίνει τιμές στις μεταβλητές τους και ταξινομεί την ουρά. Συγκεκριμένα, με ένα switch-case statement ελέγχουμε το option και στην περίπτωση, για παράδειγμα, της επιλογής του batch αλγορίθμου, προσκομίζουμε με την χρήση fscanf το όνομα του αρχείου (μεταβλητή str) και το νούμερο που αναγράφεται δίπλα σε κάθε Process (μεταβλητή num). Στην συνέχεια, τρέχουμε την newProc μία συνάρτηση που είναι υπεύθυνη για την εισαγωγή νέων processes στην ουρά. Εμβαθύνοντας, παρατηρούμε πως δεσμεύουμε δυναμικά χώρο για κάθε process και το βάζουμε στο τέλος της ουράς θέτοντας τους κατάλληλους pointers εκεί που πρέπει να δείχνουν κάθε φορά. Αυτή η διαδικασία τρέχει μέχρι να βρούμε χαρακτήρα EOF. Στην περίπτωση του Batch λοιπόν, χρησιμοποιώντας την strcpy, θέτουμε το όνομα ίσο με την μεταβλητή str, το state της διεργασίας ίσο με Ready, καθώς και το arrival time ίσο με num. Αξίζει να αναφερθεί πως σε κάθε περίπτωση, το num χειρίζεται διαφορετικά, όπως για παράδειγμα στην περίπτωση της επιλογής του αλγορίθμου SJF, όπου θέτουμε το burst time της κάθε διεργασίας ίσο με το num. Τέλος, για τους BATCH και SJF εκτελούμε bubble_sort αλγορίθμους (με τις bubble_sjf και bubble_batch) ώστε να ταξινομήσουμε σε κάθε περίπτωση τις διεργασίες με βάση το arrival time και burst time αντίστοιχα.

Γυρίζοντας πίσω στην main, ανάλογα με τον αλγόριθμο που έχουμε επιλέξει τρέχει και η κατάλληλη συνάρτηση. Για τους αλγορίθμους Batch και SJF χρησιμοποιούμε την ίδια συνάρτηση (batch_sjf) η οποία παίρνει σαν όρισμα την ουρά queue1.

Μέσα στην συνάρτηση αυτή, αρχικοποιούμε μεταβλητές τύπου timespec για την χρονομέτρηση των διεργασιών και μία μεταβλητή τύπου proc* ώστε να κρατάμε την διεργασία η οποία πρόκειται να εκτελεστεί.

Μπαίνοντας στο while βρόγχο, μέχρι να αδειάσει η λίστα, εκτελούμε τα εξής βήματα:

- Βγάζουμε από την λίστα την διεργασία που πρόκειται να εκτελέσουμε με την χρήση της deQueue η οποία μας επιστρέφει έναν pointer στην διεργασία αυτή.
- Αρχίζουμε την χρονομέτρηση της διεργασίας.
- Κάνουμε fork.
- Ο γονέας θέτει το pid της συγκεκριμένης διεργασίας στο current_process.
- Ο γονέας θέτει την κατάσταση της διεργασίας σε RUNNING.
- Ταυτόχρονα με execl αρχίζει η εκτέλεση της διεργασίας-παιδί.
- Ο γονέας περιμένει να εκτελεστεί το παιδί.
- Ο γονέας θέτει την κατάσταση της διεργασίας σε EXITED.
- Η χρονομέτρηση τελειώνει και γίνονται οι κατάλληλοι υπολογισμοί.
- Εκτυπώνεται μήνυμα με το ποια διεργασία πήρε πόσο χρόνο να εκτελεστεί και πόση ώρα έχει τρέξει όλο το πρόγραμμα μετά την εκτέλεση της διεργασίας.
- Τέλος εκτυπώνεται ο συνολικός χρόνος εκτέλεσης του προγράμματος.

Ένα παράδειγμα εκτέλεσης της Batch και SJF μπορείτε να δείτε παρακάτω:

```
Batch Algorithm Selected.
process 1188 begins
process 1188 ends
PID 1188 - CMD: work7
                Elapsed time: 6.868 secs
                Workload Time: 6.868 secs.
process 1189 begins
process 1189 ends
PID 1189 - CMD: work7
                Elapsed time: 6.861 secs
                Workload Time: 13.729 secs.
process 1190 begins
process 1190 ends
PID 1190 - CMD: work7
                Elapsed time: 7.100 secs
                Workload Time: 20.829 secs.
process 1191 begins
process 1191 ends
PID 1191 - CMD: work7
                Elapsed time: 6.929 secs
                Workload Time: 27.758 secs.
process 1192 begins
process 1192 ends
PID 1192 - CMD: work7
                Elapsed time: 6.819 secs
                Workload Time: 34.576 secs.
WORKLOAD TIME: 34.576 seconds
Scheduler exits
```

```
SJF Algorithm Selected.
process 1196 begins
process 1196 ends
PID 1196 - CMD: work7
                Elapsed time: 6.852 secs
                Workload Time: 6.852 secs.
process 1263 begins
process 1263 ends
PID 1263 - CMD: work7
                Elapsed time: 6.697 secs
                Workload Time: 13.549 secs.
process 1264 begins
process 1264 ends
PID 1264 - CMD: work7
                Elapsed time: 6.850 secs
                Workload Time: 20.399 secs.
process 1267 begins
process 1267 ends
PID 1267 - CMD: work7
                Elapsed time: 6.694 secs
                Workload Time: 27.093 secs.
process 1268 begins
process 1268 ends
PID 1268 - CMD: work7
                Elapsed time: 6.851 secs
                Workload Time: 33.944 secs.
WORKLOAD TIME: 33.944 seconds
Scheduler exits
```

Για τον αλγόριθμο Round Robin έπρεπε να υλοποιηθεί διαφορετικά η συνάρτηση του. Συγκεκριμένα, χρησιμοποιούμε τις εντολές sigemptyset και sigaction για να αρχικοποιήσουμε τον handler του SIGCHLD.

Έπειτα αρχικοποιούμε μεταβλητές τύπου timespec που θα τις χρησιμοποιήσουμε στην κλήση της συνάρτησης nanosleep και για την χρονομέτρηση των διεργασιών μας και μία μεταβλητή τύπου proc* ώστε να κρατάμε την διεργασία η οποία πρόκειται να εκτελεστεί.

Μπαίνοντας στον πρώτο while βρόγχο, μέχρι να αδειάσει η αρχική μας λίστα, εκτελούμε τα εξής βήματα:

- Βγάζουμε από την λίστα την διεργασία που πρόκειται να εκτελέσουμε με την χρήση της deQueue η οποία μας επιστρέφει έναν pointer στην διεργασία αυτή και τον θέτουμε σε έναν Process pointer (p).
- Έπειτα, βάζουμε στο τέλος μιας καινούριας λίστας το process p.

Μπαίνοντας στον δεύτερο while βρόγχο, μέχρι να αδειάσει η καινούρια μας λίστα, εκτελούμε τα εξής βήματα:

- Βγάζουμε από την λίστα την διεργασία που πρόκειται να εκτελέσουμε με την χρήση της deQueue η οποία μας επιστρέφει έναν pointer στην διεργασία αυτή.
- Αρχίζουμε τα Clocks που χρειάζονται για την χρονομέτρηση
- Θέτουμε τον global g_pointer process ίσο με το p.
- Ελέγχουμε αν η κατάσταση της είναι READY. Αν είναι:
  - Κάνουμε Fork
  - Ταυτόχρονα με execl αρχίζει η εκτέλεση της διεργασίας-παιδί.
- Εάν δεν είναι READY
  - Με την χρήση της kill στέλνεται το σήμα SIGCONT στην διεργασία και κάνει Continue.
- Θέτουμε την κατάσταση της διεργασίας σε RUNNING.
- Καλείται η nanosleep για τον χρόνο που έχει ζητήσει ο χρήστης.
- Ελέγχουμε αν η κατάσταση της είναι EXITED. Αν είναι:
  - Βάζουμε την διεργασία στην αρχική λίστα και κάνουμε continue.
- Αν δεν είναι:
  - Σταματάμε το Clock και υπολογίζουμε το χρόνο που πέρασε.
  - Τον προσθέτουμε στο burst time της διεργασίας
  - Αυξάνουμε τον temp_time και τον θέτουμε στο workload time της διεργασίας.
- Με την χρήση της kill στέλνεται το σήμα SIGSTOP στην διεργασία και κάνει Stop.
- Αλλάζουμε την κατάσταση της διεργασίας σε STOPPED
- Την βάζουμε στο τέλος της καινούριας λίστας.

Σε περίπτωση που μία διεργασία τελειώσει, στέλνει το σήμα SIGCHLD. Αυτό το σήμα το χειριζόμαστε στην συνάρτηση childHandler. Τα περιεχόμενα του handler φαίνονται παρακάτω:

```
void childHandler(int signum) {
    proc *p = g_proc;
    int status;
    waitpid(p->pid, &status, 0);
    clock_gettime(CLOCK_MONOTONIC, &end_t);
    double elapsed_time = (end_t.tv_sec - start_t.tv_sec) + (end_t.tv_nsec - start_t.tv_nsec) / 1000000000.0;
    p->bt += elapsed_time;
    p->wt += elapsed_time;
    strcpy(p->state, "EXITED");
    printf("PID %d - CMD: %s \n\t\t Elapsed time: %.3f secs \n\t\t Workload Time: %.3f secs.\n", p->pid, p->name, p->bt, p->wt);
}
```

- Όταν καλείται ο childHandler κάνουμε waitpid για την συγκεκριμένη εργασία που στέλνει το σήμα ώστε να τερματίσει. Σταματάμε το clock και υπολογίζουμε τον χρόνο που πέρασε από το προηγούμενο κβάντο μέχρι να σταματήσει η εκτέλεση της διεργασίας καθώς μπορεί να τελείωσε πριν περάσει όλο το κβάντο χρόνου. Τέλος,

Εκτυπώνεται μήνυμα με το ποια διεργασία πήρε πόσο χρόνο να εκτελεστεί και πόση ώρα έχει τρέξει όλο το πρόγραμμα μετά την εκτέλεση της διεργασίας.

- Ένα παράδειγμα εκτέλεσης της Round Robin μπορείτε να δείτε παρακάτω:

```
RR Algorithm Selected.
process 1160 begins
process 1161 begins
process 1162 begins
process 1163 begins
process 1164 begins
process 1160 ends
PID 1160 - CMD: work7
                Elapsed time: 6.810 secs
                Workload Time: 22.813 secs.
process 1161 ends
PID 1161 - CMD: work7
                Elapsed time: 6.774 secs
                Workload Time: 24.778 secs.
process 1162 ends
PID 1162 - CMD: work7
                Elapsed time: 6.742 secs
                Workload Time: 26.747 secs.
process 1163 ends
PID 1163 - CMD: work7
                Elapsed time: 6.820 secs
                Workload Time: 28.824 secs.
process 1164 ends
PID 1164 - CMD: work7
                Elapsed time: 6.826 secs
                Workload Time: 30.831 secs.
Scheduler exits
```