Κωνσταντίνος Θεοφίλης

sdi1600287

Λειτουργικά Συστήματα

Εργασία ΙΙΙ

Στην συγκεκριμένη εργασία καλούμασταν να υλοποιήσουμε ένα πρόγραμμα το οποίο παράγει διεργασίες. Η κεντρική διεργασία πρέπει να δημιουργήσει διεργασίες παιδιά (αντίγραφα της ίδιας). Όλες οι διεργασίες (πατέρας και παιδιά) «επικοινωνούσαν» μέσω ενός shared memory segment.

1. ΜΕΤΑΓΛΩΤΤΙΣΗ – ΕΚΤΕΛΕΣΗ

Για την μεταγλώττιση του προγράμματος χρησιμοποιείστε το makefile. Εντολή "make"

```
linux01:/home/users/sdi1600287/OS/os3_final_only_code>make
g++ -c -pthread -lrt feeder.cpp
g++ -c -pthread -lrt my_methods.cpp
g++ -c -pthread -lrt shared_memory.cpp
g++ -c -pthread -lrt child.cpp
g++ -pthread -lrt -o feeder feeder.cpp my_methods.cpp shared_memory.cpp child.cpp
```

Για την εκτέλεση του προγράμματος απλά χρησιμοποιείστε την εντολή:

./feeder -n <int> -M <int>

Παράδειγμα εκτέλεσης:

```
linux01:/home/users/sdi1600287/OS/os3_final_only_code> ./feeder -n 7 -M 12000
Process with ID: 13218 has average time:  16.371245 ms
Process with ID: 13222 has average time:  16.856073 ms
Process with ID: 13221 has average time:  18.482661 ms
Process with ID: 13219 has average time:  18.823266 ms
Process with ID: 13224 has average time:  18.984738 ms
Process with ID: 13223 has average time:  19.315065 ms
Process with ID: 13220 has average time:  19.482455 ms
```

2. ΟΡΙΣΜΑΤΑ – ΑΡΧΙΚΟΠΟΙΗΣΗ – ΚΟΙΝΗ ΜΝΗΜΗ

Πιο συγκεκριμένα, ξεκινάμε δίνοντας στο πρόγραμμα μας ως ορίσματα, το πλήθος των διεργασιών παιδιών που θέλουμε να δημιουργήσουμε (-n) και το πλήθος των ακεραίων που θα υπάρχουν εντός του πίνακα (-M). Εδώ να ξεκαθαρίσουμε πως η βασική διεργασία δημιουργεί έναν πίνακα με M στοιχεία (τυχαίοι ακέραιοι) ο οποίος αντιγράφεται ακριβώς όπως είναι και στις n διεργασίες παιδιά. Έν συνεχεία θα δημιουργήσουμε ένα shared memory segment στο οποίο θα μπορεί να έχει πρόσβαση κάθε διεργασία. Η δημιουργία και η πρόσβαση στην κοινή μνήμη γίνεται

με την χρήση των shmget() και shmat() αντίστοιχα. Λίγο πιο αναλυτικά, η κοινή μνήμη δημιουργήθηκε ως ένα struct. Το struct περιλαμβάνει:

```
pthread_mutex_t SHM_MUTEX;
pthread_cond_t SHM_COND_CAN_WRITE;   //Ενημερώνει την βασική διεργασία πως ένας κύκλος
                                     ολοκληρώθηκε και μπορεί να γράψει στην κοινή μνήμη.
pthread_cond_t SHM_COND_CAN_READ;    // Ενημερώνει τις διεργασίες παιδιά πως μπορούν να
                                     διαβάσουν από την κοινή μνήμη.
int current_cell;         // μετρητής που βοηθά σε έλεγχο.
int data;                 // ο ένας ακέραιος που θα αποθηκεύεται από την βασική διεργασία.
double smTimeStamp;       // η χρονοσφραγίδα.
int readersCounter;       // μετρητής που μετρά πόσες διεργασίες παιδιά έχουν διαβάσει την
                          χρονοσφραγίδα.
```

Τέλος, με την χρήση της συνάρτησης sharedMemoryInit() αρχικοποιούμε όλες τις μεταβλητές της κοινής μνήμης καθώς επίσης κάνουμε τα conditional variables και τον mutex εκτελέσιμα μεταξύ διεργασιών (PTHREAD_PROCESS_SHARED).

3. ΔΙΕΡΓΑΣΙΕΣ – WRITER – READERS

Όπως προανέφερα η κύρια διεργασία δημιουργεί διεργασίες παιδιά, πιστά αντίγραφα της. Για την αντιμετώπιση του προβλήματος της άσκησης υλοποίησα τον κώδικα Reader-Writer με πολλούς reader και έναν μόνο writer. Συνοπτικά, ο writer (κύρια διεργασία) γράφει στην κοινή μνήμη έναν ακέραιο μαζί με την χρονοσφραγίδα του. Αφού το γράψιμο ολοκληρωθεί ενημερώνει τους readers (διεργασίες παιδιά) οι οποίοι με την σειρά τους πάνε και διαβάζουν από την κοινή μνήμη την χρονοσφραγίδα που υπάρχει. Κάνουν υπολογισμούς και υπολογίζουν την χρονική καθυστέρηση και τον κινητό μέσο όρο τους. Κάθε φορά που ο writer γράψει στην κοινή μνήμη, οι readers διαβάσουν και κάνουν του υπολογισμούς τους λέμε πως ολοκληρώθηκε ένας κύκλος. Ο κύκλος τελειώνει όταν και οι n διεργασίες παιδιά διαβάσουν αυτό που ο writer έγραψε και κάνουν τους υπολογισμούς τους. Αφού τελειώσει ο κύκλος, ο writer ενημερώνεται και περνά στην κοινή μνήμη την i+1 τιμή του με νέα χρονοσφραγίδα. Η διαδικασία επαναλαμβάνεται στις n διεργασίες M φορές. Αφού ολοκληρωθούν όλοι οι κύκλοι τυπώνονται στατιστικά για τις διεργασίες στην κονσόλα. Επίσης, η τελευταία διεργασία δημιουργεί ένα αρχείο μέσα στο οποίο γράφονται στατιστικά γι' αυτή. Συγκεκριμένα, γράφουμε το PID της διεργασίας, τον κινητό μέσο όρο καθυστέρησης και την ακολουθία ακεραίων που είχε μέσα ο πίνακας. Επιπροσθέτως, αφού ολοκληρωθεί η τελευταία διεργασία γίνεται detach από την κοινή μνήμη. Μετά και από αυτή την διαδικασία το πρόγραμμα ολοκληρώθηκε. Με την χρήση της συνάρτησης destruction() αποδεσμεύουμε τον χώρο στην μνήμη και καταστρέφουμε τα conditional variables και τον mutex από την κοινή μνήμη.