Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Λειτουργικά Συστήματα
Εργασία 1

Ονοματεπώνυμο: Μαρία-Μαλαματή Παπαδοπούλου Αριθμός Μητρώου: 1115202200134

Η εργασία περιλαμβάνει τα εξής 2 αρχεία:

- erg.c, όπου περιλαμβάνει τον κώδικα
- Tekmhriwsh.pdf, όπου περιλαμβάνει την τεκμηρίωση που εξηγεί τον τρόπο με τον οποίο εργάστηκα.

Σχετικά με το αρχείο erg.c

Εισαγωγή

Ο σκοπός αυτής της εργασίας είναι η ανάπτυξη ενός προγράμματος στη γλώσσα προγραμματισμού C, το οποίο αξιοποιεί διαμοιραζόμενη μνήμη και σημαφόρους (semaphores) για τη διαχείριση και το συγχρονισμό πολλαπλών διεργασιών. Στο πλαίσιο της υλοποίησης, επιτεύχθηκε η αποτελεσματική συνεργασία μεταξύ μιας γονικής διεργασίας και πολλαπλών παιδιών-διεργασιών, χρησιμοποιώντας μηχανισμούς επικοινωνίας και συγχρονισμού που εξασφαλίζουν την ορθή εκτέλεση με διαδιεργασιακή επικοινωνία (IPC).

Δομή του προγράμματος

Το πρόγραμμα αποτελείται από τον κεντρικό κορμό (συνάρτηση main) και τρεις βοηθητικές συναρτήσεις.
Οι τρεις βοηθητικές συναρτήσεις είναι οι εξής:

- read_data: Αναλαμβάνει την ανάγνωση του text file, συγκεκριμένα του configuration file και την αποθήκευση των δεδομένων του σε δυναμικά κατανεμημένη μνήμη μέσω του struct Data, το οποίο αποτελείται από 3 πεδία, το int number που δηλώνει την ακέραια «χρονοσήμανση», το int process που δηλώνει την ετικέτα διεργασίας και το char type που δηλώνει την εντολή.
- max_i_processes: Υπολογίζει τον μέγιστο αριθμό ενεργών παιδιών-διεργασιών με βάση τα δεδομένα του configuration file.
- read_random_line: Επιλέγει και επιστρέφει μία τυχαία γραμμή από ένα αρχείο κειμένου που της έχει δωθεί σαν όρισμα.

Ο κεντρικός κορμός (συνάρτηση main)

Main

Ορθή Εκτέλεση Γραμμής Εντολών

Αρχικά, υπάρχει ένας έλεγχος για τα ορίσματα που δώθηκαν στην γραμμή εντολών του προγράμματος. Συγκεκριμένα, για να λειτουργήσει σωστά το πρόγραμμα χρειάζεται να δωθεί πρώτα το configuration file και στην συνέχεια το M σε μορφή ./erg config_3_1000.txt 5

Μεταβλητές Προγράμματος

Στην συνέχεια, θα ήθελα να εξηγήσω τι μεταβλητές χρησιμοποιήθηκαν.

- int shm_id, char *shm_text, void *shm_base: Μεταβλητές για την διαμοιραζόμενη μνήμη. Συγκεκριμένα, η int shm_id για την δημιουργία της διαμοιραζόμενης μνήμης, η char *shm_text για την τυχαία γραμμή από ένα αρχείο κειμένου και η void *shm_base για την επισύναψη της διαμοιραζόμενης μνήμης
- char *filename = argv[1], Data *data_array: Μεταβλητή για το configuration file και την δομή struct Data, όπου αποθηκεύονται δυναμικά τα δεδομένα του configuration file
- char *random_line: Μεταβλητή για την δυναμικά δεσμευμένη τυχαία γραμμή από το αρχείο κειμένου
- int num_commands: Μεταβλητή για το πλήθος των εντολών που θα εκτελέσει το parent process
- int children_pid[M+1]: Μεταβλητή για τα PID της γονικής διεργασίας και των παιδιών-διεργασιών

Καλούνται οι συναρτήσεις read_data, ώστε να αποθηκευτούν τα δεδομένα τουconfiguration file στον πίνακα data_array και max_i_processes,ώστε να βρεθεί το μέγιστο πλήθος των ενεργών παιδιών-διεργασιών.

Έλεγχος του M

Σημαντικό να αναφερθεί ότι στο πρόγραμμα υλοποιήθηκε έλεγχος για την παράμετρο M, η οποία δίνεται ως είσοδος από τη γραμμή εντολών από τον χρήστη. Συγκεκριμένα, κατά την εκτέλεση του προγράμματος υπολογίζεται το μέγιστο πλήθος των ενεργών παιδιών-διεργασιών και συγκρίνεται με το M. Εάν η τιμή του M που δόθηκε είναι μικρότερη από το μέγιστο πλήθος των ενεργών παιδιών-διεργασιών, τότε το πρόγραμμα τερματίζει εμφανίζοντας κατάλληλο μήνυμα λάθους. Αλλιώς συνεχίζει κανονικά την εκτέλεση του.

Σημαφόροι και Διαμοιραζόμενη Μνήμη

Εν συνεχεία, το πρόγραμμα πια είναι σε θέση να ξεκινήσει την εκτέλεση του και να αναπτύξει σύστημα διεργασιών με διαδιεργασιακή επικοινωνία (IPC). Για την υλοποίηση της διαδιεργασιακής επικοινωνίας χρησιμοποιήθηκαν POSIX σημαφόροι και System V Shared Memory. Σχετικά με την System V Shared Memory, αρχικοποιείται (initialize), ώστε να περιλαμβάνει χώρο για την τυχαία γραμμή αρχείου κειμένου που θα στέλνεται στα παιδιά-διεργασίες, τους M+1 σημαφόρους που θα χρησιμοποιηθούν για την διαδιεργασιακή επικοινωνία και τον ακέραιο που θα υποδεικνύει ποια διεργασία πρέπει να γίνει Terminate. Ακολούθως πραγματοποιείται attach σε διαθέσιμο χώρο και παραχωρούνται δικαιώματα ανάγνωσης και εγγραφής στη γονική διεργασία. Τέλος, θέτουμε τους κατάλληλους pointers στα αντίστοιχα σημεία, δηλαδή shm_text για την αποστολή της τυχαίας γραμμής, sem_t * sp για τους σημαφόρους και int *terminated_process για την διεργασία που πρέπει να γίνει Terminate. Έτσι, το πρόγραμμα εξασφαλίζει οργανωμένη επικοινωνία μεταξύ διεργασιών με σωστή διαχείριση μνήμης και συγχρονισμό. Στην συνέχεια, αρχικοποιούμε (initialize) τους M+1 POSIX σημαφόρους σε 0, δηλαδή σε κατάσταση blocked, τους δίνουμε πρόσβαση για διαμοιρασμό ανάμεσα στις διεργασίες και ανεβάζουμε τον σημαφόρο της γονικής διεργασίας, ώστε να μπορέσει να ξεκινήσει την εκτέλεση της. Αρχικοποιούμε το PID της γονικής διεργασίας μέσω της συνάρτησης getpid() και των παιδιών-διεργασιών με 0, το οποίο υποδεικνύει ότι έστω το παιδί-διεργασία i δεν είναι ενεργό.

Loop Εκτέλεσης

Το πρόγραμμα διαχειρίζεται τη διαδιεργασιακή επικοινωνία μέσω ενός βρόχου for-loop, όπου κάθε επανάληψη αντιπροσωπεύει μία ακέραια χρονοσήμανση. Η εκτέλεση των εντολών πραγματοποιείται μόνο όταν η τρέχουσα χρονοσήμανση συμπίπτει με αυτή που έχει οριστεί στο configuration file. Οι πιθανές εντολές είναι οι εξής: T, S.

- Εάν η εντολή είναι S:

  Τότε η γονική διεργασία ¨γεννά' μια παιδί-διεργασία σε κατάσταση blocked, καθώς περιμένει μια τυχαία γραμμή από αρχείο κειμένου και αρχικοποιεί το PID της στον πίνακα children_pid.

  Απο την άλλη, το παιδί-διεργασία που μόλις δημιουργήθηκε αρχικοποιεί την μεταβλητή int messages_count, που υπολογίζει το πλήθος των μηνυμάτων που παρέλαβε η διεργασία και την μεταβλητή int start_time, που υποδεικνύει την χρονοσήμανση εκκίνησης. Επιπλέον, κάνει wait τον σημαφόρο της, καθώς περιμένει μια τυχαία γραμμή από αρχείο κειμένου από την γονική διεργασία.

- Εάν η εντολή είναι T:

  Τότε η γονική διεργασία ελέγχει εάν το παιδί-διεργασία που πρέπει να γίνει Terminate είναι ακόμη ενεργό. Εάν είναι ακόμη ενεργό, τότε αποστέλλει μέσω της διαμοιραζόμενης μνήμης έναν ακέραιο που υποδηλώνει την εντολή τερματισμού του παιδιού διεργασίας και ένα κείμενο που δείχνει την χρονοσήμανση ολοκλήρωσης του παιδιού διεργασίας. Στη συνέχεια, ενεργοποιεί τον σημαφόρο του παιδιού-διεργασίας και περιμένει μέσω της wait τον δικό της σημαφόρο για την ολοκλήρωση τερματισμού του παιδιού-διεργασίας. Αφού ολοκληρωθεί ο τερματισμός του παιδιού-διεργασίας η σχετική πληροφορία (exit code) απορροφάται από την γονική διεργασία και ενημερώνει το children_pid του παιδιού-διεργασίας σε 0, δηλώνοντας ότι η διεργασία δεν είναι πλέον ενεργή.

  Από την άλλη, το παιδί-διεργασία λαμβάνει μέσω της διαμοιραζόμενης μνήμης τον ακέραιο που υποδηλώνει την εντολή τερματισμού και το κείμενο που δείχνει την χρονοσήμανση ολοκλήρωσης και εκτυπώνει το πλήθος των μηνυμάτων που παρέλαβε καθώς και το συνολικό πλήθος βημάτων (χρονοσήμανση ολοκλήρωσης – χρονοσήμανση εκκίνησης) για τα οποία έμεινε ενεργή. Στην συνέχεια, αποδεσμεύει τις δυναμικά δεσμευμένες μεταβλητές random_line, data_array,ενεργοποιεί τον σημαφόρο της γονικής διεργασίας, ώστε να συνεχίσει την εκτέλεση και πραγματοποιεί τερματισμό της διεργασίας.

Σε κάθε εντολή, είτε S είτε T, η γονική διεργασία στέλνει μια τυχαία γραμμή από ένα αρχείο κειμένου σε ένα τυχαίο ενεργό παιδί-διεργασία. Για να το πετύχει αυτό, χρησιμοποιεί τη μεταβλητή test, που ελέγχει αν υπάρχουν ενεργά παιδιά-διεργασίες (δηλαδή με PID διάφορο του 0). Αν υπάρχει ενεργό παιδί, επιλέγεται τυχαία ένα για την αποστολή της τυχαίας γραμμής. Η επιλογή της τυχαίας γραμμής γίνεται με τη βοηθητική συνάρτηση read_random_line, ενώ η γραμμή μεταφέρεται μέσω της διαμοιραζόμενης μνήμης. Η γονική διεργασία ενεργοποιεί τον σημαφόρο του επιλεγμένου παιδιού-διεργασίας για να γίνει η εκτύπωση της τυχαίας γραμμής, προχωρά στην επόμενη εντολή αυξάνοντας τον δείκτη του data_array, και περιμένει μέσω του σημαφόρου της (wait) να ολοκληρωθεί η εκτύπωση της τυχαίας γραμμής από το παιδί-διεργασία.

Εντολή EXIT

Όταν η εκτέλεση του προγράμματος φτάσει στην τελευταία εντολή, η οποία είναι η EXIT, η γονική διεργασία αρχικά περιμένει μέσω του σημαφόρου της (wait) να ολοκληρωθεί η εκτύπωση της τελευταίας τυχαίας γραμμής από το παιδί-διεργασία. Στην συνέχεια, προχωράει στον τερματισμό όλων των ενεργών παιδιών-διεργασιών. Ο τερματισμός όλων των ενεργών παιδιών-διεργασιών γίνεται με τον ίδιο τρόπο που περιγράφτηκε και πάνω, δηλαδή χρησιμοποιώντας τον ακέραιο terminated_process που στέλνεται μέσω της διαμοιραζόμενης μνήμης. Συνεπώς, η γονική διεργασία ενεργοποιεί τον αντίστοιχο σημαφόρο του παιδιού-διεργασίας για να ολοκληρώσει τον τερματισμό του και περιμένει μέσω του σημαφόρου της (wait) αυτόν τον τερματισμό. Τέλος, η σχετική πληροφορία τερματισμού (exit code) απορροφάται από την γονική διεργασία και ενημερώνει το children_pid του παιδιού-διεργασίας σε 0, δηλώνοντας ότι η διεργασία δεν είναι πλέον ενεργή.

Απελευθέρωση Πόρων και Αποδέσμευση Δυναμικών Μεταβλητών

Για την ορθή ολοκλήρωση του προγράμματος, είναι απαραίτητο να απελευθερωθούν όλοι οι πόροι που χρησιμοποιήθηκαν. Αρχικά, διαγράφονται οι M+1 σημαφόροι που δημιουργήθηκαν, αποσυνδέεται η διαμοιραζόμενη μνήμη (detach) και μαρκάρεται για διαγραφή μετά το πέρας της εκτέλεσης. Επιπλέον, αποδεσμεύονται οι δυναμικά δεσμευμένες μεταβλητές random_line και data_array. Έτσι, με αυτές τις ενέργειες, το πρόγραμμα ολοκληρώνεται ομαλά. Τέλος, το πρόγραμμα ελέγχθηκε διεξοδικά για διαρροές μνήμης (memory leaks) χρησιμοποιώντας το εργαλείο Valgrind με κάθε πιθανό configuration file, και διαπιστώθηκε ότι δεν παρουσιάζονται προβλήματα στην μνήμη.