Λειτουργικά Συστήματα **Ιωάννης Χουστουλάκης**
Εργασία 1 **1115202300296**

Αρχεία που παραδίδονται:

Readme.pdf

Ergasia1.c

a.txt – configuration file

Η μεταγλώττιση γίνεται με την εντολή:

gcc -g -o ergasia1 ergasia1.c -lpthread -lrt

Και εκτελείται με τις παραμέτρους:

./ergasia1 a.txt mobydick.txt 4

Ορίζεται κατ' αρχήν με define η παράμετρος MAX_CHILDREN που ελέγχει το μέγιστο αριθμό των διαφορετικών παιδιών-διεργασιών που μπορούν να υπάρχουν στο αρχείο configuration.

Ορίζονται δύο δομές, μια runningprocess που αποθηκεύει το όνομα και το PID μιας ενεργής διεργασίας, και η opcommand, που αποθηκεύει το αν και τι εκτελείται σε κάθε step της κύριας διεργασίας, αποθηκεύοντας το βήμα, το όνομα της διεργασίας που αφορά η εντολή(C-αριθμός), την εντολή("S", "T", ή "E"), και το αν υπάρχει εντολή ("Y" ή "N").

Στην main, ελέγχονται κατ' αρχήν ο αριθμός και το είδος των παραμέτρων. Απαιτείται να είναι 3 παράμετροι, οι δυο πρώτες τα ονόματα των αρχείων configuration και text και η τρίτη ο αριθμός των μέγιστων ταυτόχρονων διεργασιών, που ελέγχεται αν είναι αριθμός. Στη συνέχεια δημιουργούμε βάση του αριθμού αυτού τη μεταβλητή num_of_max_active_processes και επιχειρούμε να ανοίξουμε για ανάγνωση τα αρχεία με τα ονόματα που δώθηκαν. Εάν αυτό δεν πετύχει το πρόγραμμα τερματίζει με σφάλμα.

Δημιουργούμε στη συνέχεια τον πίνακα των σημαφόρων και τα ονόματα των σημαφόρων («/sem2_(αριθμός)»), οι οποίοι είναι Posix named, και στη συνέχεια κάνουμε Unlink τα ονόματα αυτά και κλείνουμε όλους τους σημαφόρους πριν τους ανοίξουμε, για την περίπτωση που είχαν δημιουργηθεί σε προηγούμενη εκτέλεση του προγράμματος και για κάποιο λόγο είχαν μείνει ανοικτοί ή με δεσμευμένα τα ονόματα, προκαλώντας σφάλμα.

Ανοίγουμε τους σημαφόρους των παιδιών-διεργασιών με έλεγχο σφάλματος για το αν υπάρχουν ήδη στο σύστημα (O_EXCL) και πρόσβαση εγγραφής και ανάγνωσης και για το χρήστη και για το group. Αρχικοποιούνται με τιμή 0 ώστε να μπλοκαριστούν όταν κληθεί η sem_wait. Με τον ίδιο τρόπο ανοίγουμε το σημαφόρο για τη διεργασία-γονέα.

Διαβάζουμε όλο το αρχείο κειμένου (file2) και μετράμε των αριθμό των γραμμών που περιέχει. Το ξαναδιαβάζουμε και αποθηκεύουμε στον πίνακα text_file_content το περιεχόμενό κάθε γραμμής σε διαφορετικό στοιχείο του πίνακα.

Ομοίως μετράμε τον αριθμό γραμμών-εντολών στο αρχείο configuration (file1) και αφού ελέγξω ότι όλες οι γραμμές του έχουν το σωστό formatting (αριθμός – Ci – συμβολοσειρά) αποθηκεύω τις εντολές στον πίνακα δομών commandsstruct, ο οποίος έχει ένα στοιχείο για κάθε βήμα μέχρι το τελευταίο που υπάρχει στο configuration και περιέχει τις εντολές στα βήματα που ορίζει το configuration. Εντοπίζεται ο αριθμός του τελευταίου βήματος, αν όλες οι εντολές είναι σωστές και με αύξουσα σειρά βήματος και αν η εντολή εξόδου υπάρχει πάνω από μία φορά (οπότε επιστρέφει σφάλμα).

Αποθηκεύουμε στη συνέχεια το PID της διεργασίας-γονέα και δημιουργούμε το srand για να παράξουμε τους τυχαίους αριθμούς που θα επιλέγουν στη συνέχεια τα παιδιά και τις γραμμές που θα εκτυπώνουν.

Δημιουργούμε τη shared memory, στην οποία αποθηκεύουμε την τυχαία γραμμή, έναν πίνακα που αποθηκεύει τα PID των παιδιών-διεργασιών που είναι αυτή τη στιγμή ενεργά (όπου η θέση τους στον πίνακα υποδηλώνει τον αριθμό του σημαφόρου στον οποίο αντιστοιχούν), και την εντολή και βήμα τερματισμού, που περιέχουν το PID της διεργασίας στην οποία δίνουμε εντολή να τερματίσει και το βήμα στο οποίο βρισκόταν ο γονέας-διεργασία όταν δόθηκε αυτή η εντολή (ώστε να υπολογίσει τον αριθμό των βημάτων που ήταν η διεργασία ενεργή και να τον εκτυπώσει). Αποθηκεύονται οι δείκτες που δείχνουν στις αντίστοιχες θέσεις της shared memory για τις αντίστοιχες δομές.

Στη συνέχεια ο γονέας αρχίζει να εκτελεί τα βήματα του προγράμματος βάσει του configuration. Στην αρχή του προγράμματος, όταν εκτελεστεί ένα βήμα χωρίς ακόμα να υπάρχει κάποια εντολή και συνεπώς χωρίς να είναι ενεργή κάποια διεργασία-παιδί, ο γονέας εκτελεί το βήμα της for loop χωρίς να εκτελέσει κάποια εντολή. Όταν βρεθεί εντολή, στην περίπτωση που είναι εντολή S, αυξάνεται ο αριθμός no_of_currently_running_processes κατά 1 και ελέγχεται αν ο αριθμός αυτός ξεπερνά τον μέγιστο αριθμό ενεργών διεργασιών που ορίστηκε κατά την εκτέλεση του προγράμματος, και σε αυτή την περίπτωση επιστρέφει σφάλμα. Διαφορετικά δημιουργεί την αντίστοιχη διεργασία-παιδί και κάνει sem_wait.

Η διεργασία παιδί προσθέτει στην κοινή μνήμη το PID της σε όποια θέση βρει πρώτη κενή θέση, η οποία γνωρίζουμε ότι αντιστοιχεί στον αριθμό του σημαφόρου που θα χρησιμοποιήσει. Κάνει sem_post στο σημαφόρο του γονέα ώστε να ξεμπλοκαριστεί και εκτελεί τη συνάρτηση child_process.

Η συνάρτηση child_process με το που ξεκινάει κάνει sem_wait και μπλοκάρεται μέχρι να πάρει εντολή από το γονέα. Όταν ξεμπλοκαριστεί διαβάζει από την shared memory την terminate command.

Στην περίπτωση που δεν περιέχει το PID της διεργασίας ή την τιμή -1 (που είναι εντολή τερματισμού όλων των παιδιών), διαβάζει από την shared memory επίσης την γραμμή που επιλέχθηκε τυχαία από το κείμενο και την τυπώνει. Στη συνέχεια κάνει sem_post στο γονέα και μπλοκάρεται.

Εάν λάβει εντολή τερματισμού, εκτυπώνει τον αριθμό των βημάτων που ήταν ενεργή η διεργασία (που υπολογίζεται από τιμή που αποθηκεύεται στη shared memory όταν δοθεί

εντολή τερματισμού) και τον αριθμό μηνυμάτων που έχει λάβει, ξεμπλοκάρει το γονέα και τερματίζει.

Όταν ο γονιός ξεμπλοκαριστεί από τη sem_wait που τον μπλόκαρε όταν ξεκίνησε το παιδί, ή όταν υπάρχει έστω και μία ενεργή διεργασία παιδί και δεν υπάρχει εντολή, διαλέγει τυχαία μια διεργασία-παιδί με την συνάρτηση choose_random_active_child.

Η choose_random_active_child βρίσκει τις διεργασίες που στο βήμα που εκτελείται αυτή τη στιγμή αναμένονται να είναι ενεργές (βάσει των εντολών του configuration) και επιστρέφει μια από αυτές τυχαία. Βρίσκει το PID της και τον αντίστοιχο σημαφόρο, διαλέγει μια τυχαία γραμμή από το αρχείο κειμένου και κάνει sem_post στον σημαφόρο του παιδιού και μπλοκάρεται μέχρι το παιδί να εκτυπώσει τη γραμμή.

Στην περίπτωση που σε κάποιο βήμα πάρει την εντολή T για κάποια διεργασία, ο γονιός ελέγχει κατ' αρχήν αν όντως η διεργασία αυτή είναι ενεργή. Εάν όχι, η εντολή αγνοείται. Διαφορετικά αντιγράφεται στη shared memory το PID του παιδιού terminate command καθώς και το τρέχον βήμα στο terminate step, μειώνει το no_of_currently_running_processes κατά 1, δηλώνει στη shared memory ότι ο αντίστοιχος σημαφόρος πρόκεται να γίνει διαθέσιμος και εκτελέι sem_post στην διεργασία-παιδί ώστε να εκτελεστεί η εντολή τερματισμού και ο γονιός μπλοκάρεται μέχρι να ολοκληρωθεί η διαδικασία.

Εάν ο γονιός λάβει διαφορετική εντολή από S ή T, εφόσον είχε ελεγχθεί προκαταρκτικά ότι η μόνη άλλη έγκυρη εντολή είναι η EXIT, βγαίνει από τη for loop που εκτελείται για κάθε βήμα και εκτελεί τη διαδικασία τερματισμού ολόκληρου του προγράμματος.

Η διαδικασία αυτή δίνει στη shared memory την εντολη terminate command = -1 και στη συνέχεια κάνει sem_post σε όλους τους σημαφόρους των παιδιών, πράγμα που θα δώσει εντολή σε όλα τα παιδιά τα οποία είναι ακόμα ενεργά (διότι δεν προηγήθηκε εντολή T για αυτά) να τερματίστουν. Στη συνέχεια κλείνουν όλοι οι σημαφόροι και γίνεται unlink στα ονόματά τους. Όμοια και για το σημαφόρο του γονέα. Γίνεται unlink στο όνομα της shared memory, unmap και close στην ίδια. Κλείνουν τα ανοικτά αρχεία του configuration και text. Στη συνέχεια τερματίζεται ο γονέας και επιστρέφει 0.