ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Λειτουργικά Συστήματα Υπολογιστών

6ο εξάμηνο, Ακαδημαϊκή περίοδος 2023-2024

1η Εργαστηριακή Άσκηση
Κλήσεις συστήματος, διεργασίες και διαδιεργασιακή επικοινωνία

Εργαστήριο Υπολογιστικών Συστημάτων Ε.Μ.Π.

Φεβρουάριος 2024

1 Ανάγνωση και εγγραφή αρχείων στη C και με τη βοήθεια κλήσεων συστήματος (30%)

Στο φάκελο `/home/oslab/code/char-count` μπορείτε να βρείτε κώδικα σε C (και Makefile για το σύνολο της άσκησης) που αναζητά πόσες φορές εμφανίζεται ένας χαρακτήρας μέσα σε ένα αρχείο εισόδου και γράφει το αποτέλεσμα σε ένα αρχείο εξόδου. Τα αρχεία και ο χαρακτήρας προς αναζήτηση δίνονται από το χρήστη στη γραμμή εντολών.

Επισήμανση: Παρατηρήστε ότι κατά την κλήση συναρτήσεων με αβέβαιη κατάληξη (εδώ `fopen`) χρησιμοποιούμε κώδικα που χειρίζεται πιθανά λάθη. Ακολουθήστε αυτή την τακτική σε όλες τις ασκήσεις από εδώ και πέρα.

Ζητούμενο: Υλοποιήστε πρόγραμμα με την ίδια ακριβώς λειτουργικότητα κάνοντας όμως χρήση system calls, χωρίς δηλαδή τη χρήση της βιβλιοθήκης της C. Hints: `man 2 open`, `man 2 read`, `man 2 write`, `man 2 close`, `man 2 exit`.

Προαιρετικά: Στον κώδικα που σας δόθηκε δημιουργείται πρόβλημα αν ο χρήστης δεν δώσει τις παραμέτρους στη γραμμή εντολών σωστά. Επεκτείνετε το πρόγραμμά σας ώστε να γίνονται οι απαραίτητοι έλεγχοι και να δίνονται κατάλληλες οδηγίες χρήσης.

2 Δημιουργία διεργασιών (30%)

Ζητούμενο: Επεκτείνετε το πρόγραμμα του Ερωτήματος 1 διαδοχικά ως εξής:

1. Δημιουργήστε μία διεργασία παιδί που θα χαιρετάει τον κόσμο αναφέροντας το αναγνωριστικό της και το αναγνωριστικό του γονέα της. Ο γονέας θα τυπώνει το αναγνωριστικό του παιδιού και θα περιμένει τον τερματισμό του. Hint: `man fork`, `man wait`, `man getpid`, `man getppid`.
2. Ορίστε μια μεταβλητή, έστω *x*, στο γονέα πριν τη δημιουργία του παιδιού, αναθέστε διαφορετική τιμή στο γονέα και στο παιδί και τυπώστε την τιμή της. Τι παρατηρείτε;
3. Αναθέστε στη διεργασία παιδί την αναζήτηση του χαρακτήρα στο αρχείο (όχι όμως τον υπόλοιπο χειρισμό των αρχείων).
4. Δημιουργήστε μία διεργασία παιδί που θα εκτελεί τον κώδικα που σας δόθηκε στο Ερώτημα 1. Hint: `man execv`.

3 Διαδιεργασιακή επικοινωνία (30%)

Ζητούμενο: Επεκτείνετε το πρόγραμμα του Ερωτήματος 2 ώστε να δημιουργεί P διεργασίες παιδιά (το P μπορεί να είναι ορισμένο σαν σταθερά στο πρόγραμμά σας) οι οποίες θα αναζητούν παράλληλα το χαρακτήρα στο αρχείο και η γονεϊκή διεργασία θα συλλέγει και θα τυπώνει το συνολικό αποτέλεσμα. Όταν το πρόγραμμά σας θα δέχεται Control+C από το πληκτρολόγιο (δηλαδή το σήμα `SIGINT`) θα πρέπει αντί να τερματίζει, να τυπώνει το συνολικό αριθμό διεργασιών που αναζητούν το αρχείο.

4 Εφαρμογή παράλληλης καταμέτρησης χαρακτήρων (10%)

Ζητούμενο: Υλοποιήστε μια ολοκληρωμένη εφαρμογή παράλληλης καταμέτρησης χαρακτήρων σε ένα αρχείο. Η εφαρμογή θα δέχεται από το χρήστη το όνομα του αρχείου και το χαρακτήρα προς αναζήτηση και κατά τη διάρκεια της εκτέλεσής της θα μπορεί να δέχεται από το χρήστη εντολές για: α) πρόσθεση/αφαίρεση εργατών (διεργασιών) αναζήτησης, β) παρουσίαση πληροφορίας σε σχέση με τους εργάτες που συμμετέχουν στην αναζήτηση και γ) ενημέρωση σχετικά με την πρόοδο της αναζήτησης (ποσοστό ολοκλήρωσης εργασίας και αριθμό χαρακτήρων που έχουν βρεθεί μέχρι στιγμής). Η εφαρμογή θα πρέπει να είναι δομημένη (modular) και να βασίζεται σε τρία ξεχωριστά και ανεξάρτητα δομικά στοιχεία ως εξής:

- **Front-end**: Διεργασία που αναλαμβάνει την επικοινωνία με το χρήστη, λαμβάνει τις εντολές του και τις χειρίζεται κατάλληλα, είτε εξυπηρετώντας τις είτε προωθώντας τις στον dispatcher (βλ. συνέχεια).
- **Dispatcher**: Διεργασία που είναι υπεύθυνη για την κατανομή της εργασίας στους εργάτες (workers, βλ. συνέχεια). Ο dispatcher δέχεται εντολές από το front-end για την πρόσθεση ή αφαίρεση εργατών, μοιράζει τη δουλειά στους εργάτες, συλλέγει τα μερικά αποτελέσματα από αυτούς και επιστρέφει το συνολικό αποτέλεσμα στο front-end.
- **Worker**: Το πρόγραμμα θα πρέπει να έχει δυναμικά μεταβλητό αριθμό από workers. Κάθε worker δέχεται επαναληπτικά ένα μέρος του αρχείου προς αναζήτηση (θέση σε bytes όπου ξεκινά η αναζήτηση και αριθμός bytes στα οποία θα γινει η αναζήτηση) και για κάθε τέτοιο κομμάτι δουλειάς επιστρέφει το αποτέλεσμα στον dispatcher.

Επιπλέον χαρακτηριστικά:

- Με βάση τα παραπάνω και για να μπορεί η εφαρμογή να υποστηρίζει δυναμικό αριθμό από workers, ο dispatcher θα τεμαχίζει τη δουλειά σε πολλά μέρη, τα οποία και θα μοιράζει στους διαθέσιμους εργάτες με κυκλικό τρόπο.
- Το πρόγραμμά σας θα πρέπει να είναι ανθεκτικό στην απώλεια κάποιου worker κατά την εκτέλεση (π.χ. αν σταλεί κάποιο σήμα από το τερματικό). Αυτό σημαίνει ότι η εργασία που είχε ανατεθεί σε έναν worker του οποίου η εκτέλεση έληξε από εξωτερικά αίτια θα πρέπει να ανατεθεί σε άλλον worker και εν γένει θα πρέπει να εξασφαλιστεί η απρόσκοπτη λειτουργία της εφαρμογής. Για το συγκεκριμένο χαρακτηριστικό ο dispatcher θα πρέπει να τηρεί πληροφορίες για τις εργασίες που έχουν ανατεθεί (work pool) και τους εργάτες (worker list).

Λεπτομέρειες Υλοποίησης:

- Η επικοινωνία των μερών της εφαρμογής (front-end, dispatcher, workers) για την αποστολή/λήψη εντολών και αποτελεσμάτων θα γίνεται μέσω pipes.
- Η λειτουργία της παρουσίασης των ενδιάμεσων αποτελεσμάτων θα γίνεται με συνδυασμό σημάτων και pipes. Συγκεκριμένα, το front-end θα διακόπτει τον dispatcher με ένα σήμα και αυτός θα επιστρέφει την πληροφορία σε ένα pipe.
- Είστε ελεύθεροι/ες κατά τα λοιπά να προχωρήσετε σε όποιες σχεδιαστικές επιλογές κρίνετε σκόπιμο.

Επισημάνσεις:

- Για την κατάλληλη χρήση των σωληνώσεων για αποστολή/λήψη δεδομένων (εντολών, αποτελεσμάτων, κλπ) εξοικειωθείτε με την αποστολή και λήψη byte streams σε μία σωλήνωση, με τη χρήση της sprinft, με τους μορφοποιητές `%d`, `%5d`, `%05d` (όπου 5 τυχαίο παράδειγμα), κλπ.
- Πιθανόν να σας χρειαστεί η δυνατότητα να καλείτε την κλήση συστήματος read χωρίς αυτή να μπλοκάρει. Αναζητήστε πληροφορίες για το πως γίνεται αυτό.

- Αναζητήστε πληροφορίες για το τι συμβαίνει αν διακοπεί η εκτέλεση ενός system call (π.χ. read ή write) και τι συμβαίνει όταν το άλλο άκρο μιας σωλήνωσης στην οποία μια διεργασία πάει να γράψει ή να διαβάσει ανήκει σε διεργασία που έχει τερματιστεί.
- Αναζητήστε παραδείγματα χρήσης της execv με ταυτόχρονο πέρασμα παραμέτρων στη διεργασία παιδί.
- Πιθανότατα θα χρειαστεί να προσθέσετε κάποιες τεχνητές καθυστερήσεις (π.χ. με τη βοήθεια, των `sleep`, `usleep`) στον dispatcher και τους workers για να έχετε εποπτεία της εκτέλεσης σε πραγματικό χρόνο.