Λειτουργικά Συστήματα – Τμήμα Περιττών 2018

Εργασία 2

ΜΑΡΑΓΚΟΖΑΚΗΣ ΓΕΩΡΓΙΟΣ – 1115201500089

Δομή Εργασίας:

main_root.c, main_splitter_merger.c, main_searcher.c, functions.c, functions.h, Makefile

main_root.c : Αρχικά ελέγχονται τα ορίσματα που έχουν δοθεί από τη γραμμή εντολών, με τη βοήθεια της **readcommandline**, ώστε να ικανοποιείται η συνθήκη για οποιαδήποτε σειρά των ορισμάτων. Ύστερα υπολογίζεται το πλήθος των searchers, ανάλογα τον βάθος του δέντρου και το μέγεθος του δοθέντος αρχείου. Στη συνέχεια, ανοίγουμε ένα pipe και μέσω της fork φτιάχνουμε μια νέα **διεργασία-παιδί**, στην οποία δίνουμε πρόσβαση στο pipe, μέσω της dup2. Τέλος καλούμε την execlp, στην οποία δίνουμε μια λίστα από ορίσματα.
Όσον αφορά τη **διεργασία-πατέρας**, διαβάζουμε όσο υπάρχουν διαθέσιμα δεδομένα μέσω του pipe και τα αποθηκεύουμε σε έναν δυναμικό πίνακα, ο οποίος αυξάνεται όσο το μέγεθος των δεδομένων που "έρχονται" από το pipe. Επιπλέον, το τελευταίο δεδομένο που λαμβάνουμε είναι ένα struct που αφορά τους χρόνους των splitters_mergers και των searchers. Στη συνέχεια, γράφουμε τα δεδομένα σε ένα ascii αρχείο και δημιουργούμε με τη fork μια νέα διεργασία-παιδί, όπου στην οποία καλούμε την execlp, δίνοντας της τα κατάλληλα ορίσματα για να κληθεί η sort και να ταξινομηθεί το αρχείο. Τέλος τυπώνουμε τους χρόνους min, max, avg, για τους splitter_mergers και τους searchers, και αποδεσμεύουμε τη μνήμη.

main_splitter_merger.c : Αφού ληφθούν τα ορίσματα και μειωθεί το βάθος, καλείται η ΓΙΓΑΣ – συνάρτηση make_new_process, η οποία αναλαμβάνει όλη τη δουλειά μέχρι το δέντρο να κατέβει σε βάθος = 1 και να κληθούν οι searchers. Τέλος με τ ην επιστροφή της make_new_process, προσθέτονται οι χρόνοι των παιδιών της συγκεκριμένης, με το χρόνο αυτής και γράφονται στο pipe προς τον πατέρα της.

functions.c :

Times make_new_process(int height, char *argv[]) : Ανάλογα το ύψος που βρισκόμαστε και μέσω των ορισμάτων υπολογίζεται η αρχική και τελική εγγραφή, καθώς και το μέγεθος των εγγραφών που αντιστοιχεί στις δύο διεργασίες-παιδιά αυτής(start, left_finsh, right_start, finish, size). Στη συνέχεια, δημιουργείται ένα pipe και καλείται η fork και φτιάχνεται μια νέα διεργασία-παιδί(αριστερό παιδί), για την οποία ελέγχεται για το -s, και προσαρμόζονται ανάλογα τα όρια των εγγραφών του αρχείου για τα οποία η διεργασία αυτή είναι υπεύθυνη. Ύστερα, δίνεται πρόσβαση στο pipe για γράψιμο μέσω της dup2, και καλείται η exclp, με τα κατάλληλα ορίσματα για το εκτελέσιμο main_splitter_mergers ή αν το διαθέσιμο βάθος είναι 1 τότε καλείται το

εκτελέσιμο main_searchers. Όσον αφορά τη διαδικασία πατέρας, με τον ίδιο τρόπο όπως παραπάνω, δημιουργεί μια νέα διεργασία-παιδί(δεξιό παιδί), το οποίο ομοίως όπως παραπάνω επικαλύπτεται μέσω της execlp με το main_splitter_merger ή αν το διαθέσιμο βάθος είναι 1 τότε καλείται το εκτελέσιμο main_searcher. Τέλος η διαδικασία-πατέρας, αναλαμβάνει να διαβάσει από τα παιδιά του, searchers ή splitter_mergers, τα αποτελέσματα της αναζήτησης μέσω του pipe, τα οποία και προωθούνται μέσω του write, στο παραπάνω επίπεδο. Το διάβασμα εδώ γίνεται "παράλληλα" με την εκτέλεση των παρακάτω παιδιών-διαδικασιών, δηλαδή όσο υπάρχουν διαθέσιμα αποτελέσματα από το αριστερό και δεξιό παιδί διαβάζονται. Τα τελευταία δεδομένα που διαβάζονται είναι τα struct που αφορούν τους χρόνους εκτέλεσης των searchers, και splitter_mergers. Τέλος, ελέγχεται αν έχουν τελειώσει οι διαδικασίες παιδιά και επιστρέφονται στη main_splitter_merger οι χρόνοι για να προωθηθούν παραπάνω.

call_strstr : Ελέγχεται ένα string με ένα substring, μέσω της strstr, και επιστρέφεται ακέραιος επιτυχίας ή αποτυχίας αντίστοιχα.

check_for_substring : Για όλα τα στοιχεία της εγγραφής ελέγχεται με το pattern, μέσω της call_strstr. Αν βρεθεί το πολύ σε μία εγγραφή τότε επιστρέφεται αριθμός επιτυχίας.

read_from_bin_file : Ανοίγεται το αρχείο, και διαβάζεται με τη χρήση της read, η υλοποίηση μου διαφέρει από του κ. Δελή, καθώς χρησιμοποιώ low level I/O. Για κάθε εγγραφή που διαβάζεται ελέγχεται αν περιέχεται σε αυτή το pattern, με τη χρήση της check_for_substring. Αν βρεθεί, τότε η εγγραφή γράφεται στο pipe, μέσω της write.

Calc_The_Times : Υπολογίζονται οι min και max χρόνοι του struct των χρόνων.

main_searcher : Αφού ληφθούν τα ορίσματα, καλείται η read_from_bin_file, η οποία αναλαμβάνει το έργο της ανάγνωσης και αναζήτησης του αρχείο, καθώς και της προώθησης των αποτελεσμάτων μέσω του pipe στη διαδικασία-πατέρα της searcher. Τέλος, γράφονται μέσω του pipe ο χρόνος για τον οποίο έτρεξε η υπάρχουσα διεργασία.

Δομή αποθήκευσης χρόνων MIN-MAX:

```
typedef struct {                    Χρόνοι MIN, MAX
   double min;
   double max;
} Min_Max_Times;

typedef struct {
   Min_Max_Times sp_mg;     Αφορά τους χρόνους των splitter_mergers
   Min_Max_Times se;        Χρόνοι των παιδιών-διαδικασιών searchers
} Times;
```

Σχόλια για την υλοποίηση, την εκτέλεση και περιορισμοί:

Η υλοποίηση μου κατά την εκτέλεση για όλα τα αρχεία εμφάνιζε σωστά και ταξινομημένα όλα τα θεμιτά αποτελέσματα και δεν παρουσίαζε memory leaks κατά την εφαρμογή της valgrind. Επιπλέον, έχει υλοποιηθεί Makefile, το οποίο υποστηρίζει separate compilation.

Από τα ζητούμενα της εκφώνησης δεν έχω υλοποιήσει τη λειτουργία που αφορά την αποστολή και λήψη σημάτων SIGUSR2, καθώς και τον υπολογισμό του Turnarround Time.

Ενδεικτική εντολή εκτέλεσης:

```
./main_root -h 5 -d Records1000.bin -p 1 -s
```