

## **2η ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ**

### **ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ**

**sdi1900050**

**Στυλιανός Δημητριάδης**

Παρακάτω θα γίνει επεξήγηση της υλοποίησης μου:

Περιλαμβάνονται τα παρακάτω αρχεία:

bubblesort.c

bubblesort.h

create\_array\_of\_recs.h

Makefile

Makefile2

Makefile3

mergesort.c

mergesort.h

mysort.c

mysort.h

parsing.c

parsing.h

record.h

Οδηγίες εκτέλεσης του προγράμματος:

make

make -f Makefile2

make -f Makefile3

./mysort -i votersxxx.bin -k 3 -e1 ./bubblesort -e2 ./mergesort  
(μια ενδεικτική εκτέλεση, ./bubblesort ./mergesort πρέπει να μείνουν ίδια votersxxx .bin μπορεί να αλλάξει όπως και ο αριθμός k ,ενώ μπορεί να αλλάξει και η σειρά των ορισμάτων)

Το πρόγραμμα ξεκινάει με το parsing των ορισμάτων που δίνονται από τον χρήστη. Αρχικά αν το flag είναι -i η επόμενη λέξη θα πρέπει να είναι το αρχείο που θα πρέπει να ανοιχτεί, αν δεν υπάρχει επιστρέφει μήνυμα λάθους. Αν το flag είναι -k ο χρήστης πρέπει να δώσει τον αριθμό των παιδιών που θα δημιουργηθούν από τον γονέα, αν αυτό δεν είναι αριθμός τότε μήνυμα λάθους επιστρέφεται. Τέλος υπάρχουν τα flags -e1 -e2 όπου ο χρήστης δίνει για κάθε ένα το όνομα ενός προγράμματος ταξινόμησης (έχει υλοποιηθεί ο bubblesort και ο mergesort αλγόριθμος). Αν κάποιο από τα flags λείπει επιστρέφεται μήνυμα λάθους.

Στην συνέχεια ανοίγεται το αρχείο votersxxx.bin για διάβασμα και με την βοήθεια της συνάρτησης find number of records η οποία διαβάζει κάθε record του αρχείου και για κάθε ενα από αυτά αυξάνει ένα counter, επιστρέφει το σύνολο των voters σε ένα αρχείο στην μεταβλητή numberofrecords.

Στην συνέχεια δημιουργούνται pipes για επικοινωνία μεταξύ coordinator-splitter/merger-reporter και splitter/merger. Κάθε ένα παιδί έχει και διαφορετικό pipe για να αποφευχθούν race conditions.

Ο κώδικας ξεκινάει ορίζοντας ως signal handler τον default, και ως σήμα το SIGUSR1 signal, ενώ μετράω το σύνολο των sorters που θα δημιουργηθούν για να δεσμεύσω μνήμη για τον πίνακα timer που αποθηκεύει την ώρα σε δευτερόλεπτα για την εκτέλεση του κάθε sorter.

Για κάθε παιδί ο splitter/merger-reporter και splitter/merger πρέπει να αναθέσει έναν συγκεκριμένο αριθμό από records για να γίνουν sort. Σε κάθε παιδί θα ανατεθεί 1/k records, ενώ αν περισσεύουν records θα τα ανάλαβει το πρώτο παιδί. Κάνω NumofChildren forks, κάθε παιδί θα έχει ένα beginning range και ένα ending range, δηλαδή το line που θα αρχίσει να πρέπει να ταξινομεί και το line που θα πρέπει να τελειώνει να ταξινομεί στο αρχείο. Για αρχή κλείνουν τα pipes για διάβασμα καθώς θα γίνει μόνο write, το beginning range θα είναι ο εαυτός του + τα records που εχουν ανατεθεί στο παιδί. Με το call write γίνονται update τα values των distribute records και beginning range, ενώ με την dup2 ο write end of pipe γίνεται redirect στο stdout, ώστε να μπορέσω να διαβάσω το sorted αποτέλεσμα που θα προέλθει από τους sorters καθώς και τον αριθμό των usr2 signals και τον χρόνο εκτέλεσης κάθε sorter. Τέλος ο splitter/merger καλεί την συνάρτηση create\_sorters με ορίσματα, το beginning range, τα records που πρέπει να ταξινομηθούν, τα παιδία sorters που πρέπει να

δημιουργήσει (NumofChildren-i) τα συνολικα records του αρχείου και τα ονόματα των προγραμμάτων αλγορίθμων ταξινόμησης.

Η συνάρτηση create\_sorters είναι υπεύθυνη στο να δημιουργήσει NumofChildren-i sorters παιδιά τα οποία με την χρήση της execv θα χρησιμοποιήσουν εναλάξ τους αλγόριθμους ταξινόμησης για να κάνουν sort τα επιμέρους records.

Δημιουργούνται how\_many\_sorters pipes για την επικοινωνία μεταξύ splitter/merger και sorter και την αποφυγή race condition. Για αρχή o signal handler θα πρέπει να κάνει ignore τα signals SIGUSR2 καθώς αν δωθεί ένα signal πρίν την execv και δεν το κάνω ignore δέν θα εκτελεστεί η execv και δεν έχει νόημα να δώσω σήμα μετα την execv καθώς δεν θα εκτελεστεί. Με παρόμοια λογική αναθέτω στους sorters 1/k records με το k τώρα να είναι το NumofChildren-i (δηλαδή αν είχα αναθέσει στον splitter/merger 50 records και πρέπει να δημιουργήσω 3 sorter πρέπει να ανατεθούν σε κάθε sorter 50/3 records ). Στο σώμα των παιδιών δημιουργώ τρία char arrays όπου θα περιέχουν τα arguments με τα οποία θα εκτελεστούν οι sort αλγόριθμοι(προγράμματα).Αυτο γίνεται με την χρήση της sprintf η οποία στέλνει έναν int σε char \* μορφή. Τα arguments που θα δωθούν στα sorting προγράμματα είναι τα εξής:

parameter1=records που πρέπει να γίνουν sort από τον εκάστοτε sorter,parameter2=αριθμός record αρχείου που θα πρέπει να αρχίσει να ταξινομεί ο συγκεκριμένος sorter,parameter3=αριθμός record αρχείου που θα πρέπει να σταματήσει να ταξινομεί ο συγκεκριμένος sorter,υπάρχει ένα επιπλέον argument που δίνεται στα προγράμματα και αυτό είναι το όνομα του αρχείου.

Οι νέες τιμές αυτών των μεταβλητών γίνονται update με την χρήση του write pipe end. Έπειτα δίνεται το σήμα SIGUSR2 με το kill call το οποίο γίνεται ignore,δημιουργείται ο πίνακας argv που περιέχει τα ορίσματα που θα δωθούν στο κάθε sort πρόγραμμα και άν έχουμε άρτιο αριθμό sorter καλείται με την execv το sort1 πρόγραμμα ,ενω με περιττό το sort2 πρόγραμμα.

Τα προγράμματα bubblesort και mergesort υλοποιούν τους γνωστούς αλγορίθμους ενω και τα δύο χρησιμοποιούν την συνάρτηση create\_array\_of\_recs η οποία παίρνει ως ορισμα το όνομα του αρχείου τα records που πρέπει να ταξινομηθούν ,το record number στο file που θα πρέπει να ξεκινήσει η ταξινόμηση και το record number στο file που θα

πρέπει να τελειώσει η ταξινόμηση. Η συνάρτηση διαβάζει από την αρχή του αρχείου μέχρι να φτασει τον αριθμο beginning range, συνεχίζει και διαβάζει και καταχωρεί στον πίνακα records τα records από beginning range μέχρι ending range και επιστρέφει τον τελικό πίνακα, ο οποίος δίνεται στο merge sort η bubblesort και ταξινομείται. Όταν τελειώσει ο πίνακας κάθε στοιχείο του γίνεται write στο stdout.

Ο parent των sorter θα κάνει read κάθε line από το stdout και με την σειρά του θα αποθηκέψει τα records στο Record πίνακα. Όταν όλα τα παιδιά sorters τελειώσουν θα μαζευτούν από την wait και αν το status τους είναι WIFEXITED (αφού το σήμα έγινε ignore) θα αυξηθεί ο signal usr2 counter. Τέλος ο αριθμός των usr2 signals, ο χρόνος εκτέλεσης κάθε sorter και όλος ο πίνακας Records εκτυπώνονται στο stdout.

Ο parent (coordinator-splitter/merger-reporter) θα διαβάσει τον χρόνο εκτέλεσης κάθε sorter θα τον αποθηκέψει στον πίνακα timer, θα διαβάσει το σύνολο των usr2 signals και θα το προσθέσει στο άθροισμα ενώ ακόμα θα διαβάσει κάθε record από το stdout που προέκυψε από τους sorters κάθε merger/sorter και θα το αποθηκέψει στον τελικό πίνακα Records. Ετσι θα έχει προκύψει ένας πίνακας όχι ολικα ταξινομημένος αλλά μερικά ταξινομημένος.

Όλα τα παιδιά θα μαζευτούν από τον coordinator-splitter/merger-reporter με wait και αν το status τους είναι WIFSIGNALED (αφού το σήμα δεν έγινε ignore) θα αυξηθεί ο signal usr1 counter.

Καλείται η bubblesort για να ταξινομήσει τον τελικό πίνακα και εκτυπώνεται όλος ο πίνακας με τα ταξινομημένα records, ο χρόνος εκτέλεσης κάθε sorter, ο συνολικός χρόνος εκτέλεσης όλων των sorter και το σύνολο των usr1 και usr2 signals.