Κωνσταντίνος Τσιάρας AM: 1115201500165

## Λειτουργικά Συστήματα

# 2η Προγραμματιστική Εργασία

**Μεταγλώττιση:** *make*

**Εκτέλεση:** *./myfind -h Height -d Datafile -p Pattern [-s]*

(οι σημαίες μπορούν να εμφανίζονται με οποιαδήποτε σειρά, η "-s" μπορεί να παραλειφθεί)

## Γενική Περιγραφή:

- **Root:** Η εκτέλεση ξεκινάει από το *root.c* το οποίο, αφού πραγματοποιήσει έλεγχο ορθότητας των ορισμάτων που δόθηκαν, δημιουργεί το pipe στο οποίο θα γράφει ο αρχικός Splitter-Merger και στην συνέχεια τον δημιουργεί, μέσω *fork()* και έπειτα *execl()* το εκτελέσιμό του από το παιδί που δημιουργήθηκε. Έπειτα διαβάζει επαναληπτικά από το pipe Records τα οποία αποθηκεύει σε μια λίστα έως ότου να λάβει ένα *SMStats* struct το οποίο σηματοδοτεί τη λήξη του Splitter-Merger. Έπειτα, περνάει τη λίστα που τώρα περιέχει όλα τα Records που βρέθηκαν από τους Searchers σε μια νέα διεργασία παιδί (που δημιουργεί μετά από *fork()* με την "*execlp("/usr/bin/sort", "sort", "-n", (char *) NULL);*") η οποία τυπώνει τα Records ταξινομημένα. Τέλος, τυπώνει τα συνολικά στατιστικά του προγράμματος και τερματίζει.
- **Splitter-Merger:** Τα ορίσματα που δέχεται ο Splitter-Merger είναι, με αυτή τη σειρά: όνομα Datafile, πλήθος Searchers που θα δημιουργηθούν, πρώτος Searcher για τον οποίον είναι υπεύθυνος (για τον αρχικό θα είναι 0), τελευταίος Searcher για τον οποίον είναι υπεύθυνος (για τον αρχικό θα είναι $2^{Height} - 1$), Pattern, τρέχον Height, 1 ή 0 ανάλογα αν δόθηκε ή όχι αντίστοιχα η σημαία "-s" και το PID του Root. Στη συνέχεια ο Splitter-Merger κατασκευάζει διαδοχικά 2 παιδιά διεργασίες με ένα pipe για το καθένα από αυτά και ανάλογα με το αν το Height είναι μεγαλύτερο ή ίσο με 1 κάνει *execl()* σε καθένα από αυτά είτε πάλι το εκτελέσιμο του Splitter-Merger με το Height μειωμένο κατά ένα και με το μισό εύρος υπεύθυνων Searchers, είτε το εκτελέσιμο του Searcher. Σε κάθε περίπτωση, διαβάζει τα Records που του επιστρέφουν τα παιδιά του μέσω των pipes και τα γράφει κι αυτός με τη σειρά του στο stdout (η διαδικασία αυτή εξηγείται περαιτέρω στο «Reading & Writing» παρακάτω). Όταν λάβει στατιστικά και από τα 2 παιδιά του τότε τα συνδυάζει με τα δικά του σε ένα *SMStats* struct, το γράφει κι αυτό στο stdout και τερματίζει.
- **Searcher:** Τα ορίσματα που δέχεται ο Searcher είναι, με αυτή τη σειρά: όνομα Datafile, αριθμός πρώτου Record που θα πρέπει να ψάξει, αριθμός τελευταίου Record που θα πρέπει να ψάξει, Pattern και το PID του Root. Για κάθε Record που βρίσκεται εντός του εύρους που έλαβε από τα ορίσματά του, ελέγχει όλα του τα πεδία για την ύπαρξη του ζητούμενου Pattern και, εάν αυτό βρεθεί, το γράφει στο stdout. Αφού τα εξετάσει όλα, γράφει και τα στατιστικά του με μια δομή SearcherStats στο stdout, αποστέλλει σήμα SIGUSR2 στο Root και τερματίζει.

## Σχόλια:

### ❖ Reading & Writing:

- Η επικοινωνία μεταξύ δύο οποιωνδήποτε διεργασιών που η μία γράφει δεδομένα και η άλλη διαβάζει γίνεται μέσω ενός απλού pipe, το WRITE_END του οποίου γίνεται *dup2()* στο STDOUT_FILENO της διεργασίας-παιδί που θα πραγματοποιεί το γράψιμο. Έτσι, ακόμα και μετά το *execl()* που πραγματοποιείται, αυτή μπορεί να γράφει στο stdout όπως αναφέρθηκε παραπάνω αλλά τα όσα γράφει να γράφονται στο pipe.
- Προτού ένας Searcher ή Splitter-Merger γράψει ένα Record ή τα στατιστικά του γράφει πρώτα έναν ακέραιο: 0 για Record, 1 για δομή στατιστικών Searcher ή 2 για δομή στατιστικών Splitter-Merger. Η διεργασία γονέας (Splitter-Merger παραπάνω επιπέδου ή ο Root) κάθε φορά που πάει να διαβάσει από το pipe διαβάζει πρώτα αυτόν τον ακέραιο και ξέρει τι να περιμένει στη συνέχεια.
- Κάθε Splitter-Merger που χρειάζεται να διαβάζει δεδομένα από 2 παιδιά κάνει χρήση *poll()* προκειμένου να ξέρει ποιανού το pipe περιέχει δεδομένα ανά πάσα στιγμή. Αυτό αποτρέπει άσκοπο μπλοκάρισμα που θα προέκυπτε απλά με διαδοχικές *read()*.
- Ως γνωστόν, οι read() και write() ενδέχεται να επιστρέψουν λιγότερα bytes απ' ότι ζητήθηκαν χωρίς όμως να έχει προκύψει κάποιο σφάλμα. Έλεγχος γι' αυτό γίνεται παντού όπου χρησιμοποιούνται μέσω των συναρτήσεων *readFromFd()* και *writeToFd()* που βρίσκονται στο *util.c* προκειμένου να διαβάζεται ολόκληρο κάθε φόρα αυτό που αναμένεται.

### ❖ Skew option:

Όταν ο Splitter-Merger φτάνει να δημιουργήσει παιδιά Searchers υπολογίζει και τους περνάει το εύρος των Records τα οποία θα πρέπει να ψάξουν. Σε περίπτωση που δεν έχει δοθεί το "-s" flag το εύρος αυτό αντιστοιχεί απλά σε $k/2^{height}$ (όπου $k$ το πλήθος των Records του Datafile) ενώ αν έχει δοθεί τότε αυτό είναι, σύμφωνα με την εκφώνηση, $k * i \sum_{n=1}^{2^{height}} n$ για τον $i$-οστό Searcher. Καθώς ο αριθμός αυτός στη δεύτερη περίπτωση ενδέχεται να μην είναι ακέραιος και προκειμένου όμως να πάρει ακέραιο πλήθος από Records, γίνεται η παραδοχή ότι ο τελευταίος Searcher θα πάρει όσα Records έχουν περισσέψει μέχρι το τέλος του Datafile, ακόμη κι αν αυτά προκύπτουν λιγότερα από τον αριθμό που έχει υπολογιστεί με τον παραπάνω τύπο.

### ❖ Signal Handling:

- Η διαχείριση των SIGUSR2 που φτάνουν στο Root γίνονται με τη χρήση sigaction() και κατάλληλου signal handler σε αυτόν πριν καν τη δημιουργία των Searchers.
- Έχει γίνει πρόβλεψη ώστε τα εισερχόμενα signals να μην διακόψουν ενδεχόμενες *read()* ή *write()* με τη χρήση του SA_RESTART flag στη *sigaction()* αλλά και με έλεγχο του errno για EINTR μετά από αποτυχία αυτών (και επανάληψή τους σε αυτή την περίπτωση).
- Σημειώνεται πως ενδέχεται κάποιο ή κάποια σήματα να χαθούν (στην περίπτωση που φτάσουν ταυτόχρονα και όσο ο signal handler προσπαθεί ακόμα να διαχειριστεί το πρώτο που του ήρθε) και έτσι το τελικό τους πλήθος να είναι μικρότερο από $2^{Height}$ όπως θα περιμέναμε. Κάτι τέτοιο είναι αναμενόμενο και δεν συνεπάγεται σφάλμα σε κάποιον από τους Searchers ή οπουδήποτε αλλού.

❖ **Στατιστικά:**

- Εκτός από τα στατιστικά που ζητούνται από την εκφώνηση έχω προσθέσει το πλήθος των Records που εξετάστηκαν (συνολικά και ανά Searcher), το πλήθος των Records που βρέθηκαν να ταιριάζουν με το Pattern (συνολικά και ανά Searcher) και το χρόνο της διεργασίας Sorter. Τα πλήθη Records ανά Searcher συγκεκριμένα έκρινα πως είναι ιδιαιτέρως χρήσιμα για επισκόπηση της λειτουργίας του προγράμματος όταν έχει δοθεί το "-s" flag.
- Σε όλες τις μετρήσεις χρόνων γίνεται χρήση πραγματικού χρόνου ("clock" time και όχι CPU time) μέσω της *gettimeofday()*.

## Άλλες Πρόσθετες Λειτουργία:

- *IGNORE_CASE*: Από προεπιλογή, οι Searchers ψάχνουν να βρουν Records που περιέχουν το δοθέν Pattern **ακριβώς** όπως δίνεται. Προσφέρεται η δυνατότητα αναζήτησής του αγνοώντας κεφαλαία/πεζά γράμματα θέτοντας τη σταθερά *IGNORE_CASE* σε *true* στο *util.h* .
- *PRETTY_PRINT_RESULTS*: Λαμβάνοντας υπόψη τα αναμενόμενα Records με βάση τα αρχεία που μας δόθηκαν, υλοποίησα έναν «όμορφο» τρόπο παρουσίασής τους με τη μορφή πίνακα. Αυτός μπορεί να απενεργοποιηθεί και κάθε Record να εμφανίζεται απλά σε μια γραμμή με τα πεδία να χωρίζονται με κενά, αλλάζοντας τη σταθερά *PRETTY_PRINT_RESULTS* σε *false* στο *util.h* .