Κωνσταντίνος Τσιάρας AM: 1115201500165

Προγραμματισμός Συστήματος

Εργασία 2η

Μεταγλώττιση: $make$

Εκτέλεση: i) $./jobExecutor\ -d\ docfile\ -w\ numWorkers$

ii) $./jobExecutor\ -w\ numWorkers\ -d\ docfile$

όπου $docfile$ το όνομα του αρχείου κειμένων και w ένας θετικός ακέραιος

Επιλογές Υλοποίησης:

❖ Χρησιμοποιούμενες Δομές:

- Docs: Τα κείμενα όλων των φακέλων που διαχειρίζεται ένας worker αποθηκεύονται σειριακά σε κάθε worker σε μία δομή *char**** έτσι ώστε να διακρίνονται οι ξεχωριστές γραμμές μέσα σε κάθε κείμενο. Για παράδειγμα, το *docs[6][9][42]* επιστρέφει τον 42ο χαρακτήρα της 9ης γραμμής του 6ου (συνολικά) κειμένου που διαχειρίζεται ο συγκεκριμένος worker.
- Inverted Index: Υλοποιείται με ένα Trie, όπως δημιουργήθηκε για την 1η εργασία: Κάθε κόμβος του Trie (*TrieNode*) περιλαμβάνει ακριβώς ένα χαρακτήρα, δείκτη *next* σε *TrieNode* του ίδιου επιπέδου, δείκτη *child* σε *TrieNode* του παρακάτω επιπέδου και έναν δείκτη σε *PostingList*. Σε κάθε επίπεδο του Trie διατηρούνται πάντα ταξινομημένοι.
- PostingList: Το PostingList διαθέτει απλά έναν δείκτη στο πρώτο και στο τελευταίο PostingListNode του συγκεκριμένου TrieNode όπου, σε αντίθεση με την 1η εργασία, εδώ αποθηκεύονται το id του κειμένου στο οποίο εμφανίζεται η συγκεκριμένη λέξη, το term frequency (*tf*) αυτής όπως και μια λίστα ακεραίων με τις γραμμές ακριβώς στις οποίες εμφανίζεται.

❖ Pipes:

- Τα pipes κατασκευάζονται στον φάκελο "./pipes" χρησιμοποιώντας τη σύμβαση "./pipes/Worker*N*_*P*" όπου *N* ο αριθμός του worker και *P*={0,1} για τα pipe που ο WorkerN διαβάζει και γράφει αντίστοιχα και διαγράφονται κατά τον τερματισμό του προγράμματος. Τα logfiles κατασκευάζονται στον φάκελο "./log" χρησιμοποιώντας τη σύμβαση "./log/Worker*N*/*worker_pid*.log" καθώς για κάθε worker μπορεί να έχουν υπάρξει παραπάνω από μία διεργασίες σε μία εκτέλεση του προγράμματος. Τα logfiles διατηρούνται και μετά τον τερματισμό του προγράμματος εκτός εάν η έξοδος γίνει με τη χρήση "*/exit -l*". Τα παραπάνω paths είναι ορισμένα στο *path.h* και δύναται εύκολα να αλλάξουν.

- Με εξαίρεση τα paths των φακέλων που περνάει η jobExecutor στους workers κατά τη δημιουργία των δεύτερων (τα οποία είναι σταθερού μεγέθους *PATH_MAX*), όπου αλλού απαιτείται πέρασμα πληροφορίας μεταξύ jobExecutor και worker και αντιστρόφως μέσω κάποιου pipe αυτό γίνεται περνώντας πρώτα απ' το pipe έναν ακέραιο που δηλώνει το μέγεθος του επόμενου μηνύματος ώστε ο παραλήπτης να ξέρει πόσο να διαβάσει με τη *read()*.
- Για αποδοτικό διάβασμα από πολλά pipes (συγκεκριμένα για το διάβασμα απαντήσεων από τους workers στην jobExecutor αφότου αυτή τους στείλει οποιοδήποτε query) γίνεται χρήση της συνάρτησης *poll()*.
- Στα σημεία όπου απαιτείται μεταφορά μηνυμάτων απροσδιόριστου από πριν πλήθους, προστίθεται στο τέλος ένα ακόμα μήνυμα "$" που σημαίνει πως δεν ακολουθούν άλλες απαντήσεις για το συγκεκριμένο ερώτημα.

❖ Signals:

- Σε περίπτωση εκπνοής του deadline για κάποιο search query - υπολογίζεται στην jobExecutor με τη χρήση *alarm()* - αυτή ενημερώνει τους workers που δεν έχουν ακόμα τερματίσει την αναζήτηση να την σταματήσουν απρόσμενα με τη χρήση σήματος *SIGUSR1* και αγνοεί για το τελικό αποτέλεσμα τυχόντα αποτελέσματα που εκείνοι είχαν επιστρέψει μέχρι εκείνο το σημείο.
- Οποτεδήποτε η jobExecutor λάβει κάποιο διαχειρίσιμο σήμα τερματισμού (*SIGINT, SIGTERM, SIGQUIT*) τότε, αφού ολοκληρώσει την εντολή που τυχόν εκτελεί εκείνη τη στιγμή, στέλνει SIGTERM στους workers και τερματίζει αφού απελευθερώσει ό,τι μνήμη είχε δεσμεύσει. Αντίστοιχα, όταν κάποιος worker λάβει κάποιο από τα παραπάνω σήματα (είτε από την jobExecutor είτε από τον χρήστη) τότε επίσης, αφού ολοκληρώσει ό,τι εκτελεί εκείνη τη στιγμή, απελευθερώνει ό,τι μνήμη είχε δεσμεύσει και τερματίζει.
- Αν κάποιος worker τερματίσει οποιαδήποτε στιγμή χωρίς αυτό να οφείλεται σε τερματισμό της jobExecutor (π.χ. από εντολή Linux "kill *worker_pid*" που ενδεχομένως έτρεξε ο χρήστης) τότε η τελευταία μέσω διαχείρισης του σήματος *SIGCHLD* φτιάχνει καινούργιο worker περνώντας του τους φακέλους που διαχειρίζονταν αυτός ο οποίος τερματίστηκε απρόσμενα.

❖ Logging:

- Το logging για κάθε είδους query από τους workers γίνονται ως εξής:
  /search: *Time* : search : *keyword1* : *pathname1* : *pathname2* : *pathname3* : … : *pathnameN*
  Time : search : *keyword2* : *pathname1* : *pathname2* : *pathname3* : … : *pathnameN*
  …
  /maxcount: *Time* : maxcount : *keyword* : *pathname* (*count*)
  /mincount: *Time* : mincount : *keyword* : *pathname* (*count*)
  /wc: *Time* : wc : *byte_count* : *word_count* : *line_count*
- Logging γίνεται από κάθε worker ακόμη κι αν το ζητούμενο keyword δεν βρίσκεται στα αρχεία του. Σε αυτή την περίπτωση στο logfile τυπώνεται απλά "*Time* : query : *keyword* : ".

❖ **Bash Scripts:**

- Το πρώτο script εκτυπώνει, όπως περιγράφει η εκφώνηση, το πλήθος των search queries - και συγκεκριμένα, keywords - που **συνολικά** έλαβαν οι workers (ανεξάρτητα από το αν βρήκαν αποτελέσματα για αυτά στα αρχεία τους). Η απαιτούμενη αλλαγή για να εκτυπώνει το πλήθος μόνο όσων απέδωσαν αποτελέσματα περιέχεται σε σχόλιο εντός του.
- Στα 2 τελευταία bash scripts, σε περίπτωση ισοβαθμίας (περισσότερες από μια λέξεις εμφανίζονται τον μέγιστο/ελάχιστο αριθμό φορών) τότε εκτυπώνονται όλες αυτές.

❖ **Διάφορα:**

- Οι φάκελοι μοιράζονται απλά στους workers, με την ανάθεση να γίνεται ως εξής:
  Worker1: Φάκελοι 1, w+1, 2*w+1, …
  Worker2: Φάκελοι 2, w+2, 2*w+2, …
  …
  WorkerW: Φάκελοι w, 2*w, 3*w, …
  Προφανώς, σε περίπτωση που το πλήθος των φακέλων δεν διαιρείται ακριβώς με αυτό των workers τότε κάποιοι απ' τους πρώτους workers θα διαχειρίζονται έναν παραπάνω φάκελο από τους υπόλοιπους.
- Σε περίπτωση που το w που δίνεται από τον χρήστη υπερβαίνει το πλήθος των φακέλων τότε τίθεται αυτομάτως ίσο με αυτό.
- Το *docfile* δύναται να περιέχει τόσο σχετικά όσο και απόλυτα paths. Εσωτερικά στο πρόγραμμα αποθηκεύονται όλα ως απόλυτα και χρησιμοποιούνται παντού (εκτυπώσεις και logging) με αυτή τη μορφή.
- Υποστηρίζεται αναζήτηση και άνω των 10 keywords ανά search query.

Παραδοχές:

- Το μήκος των μονοπατιών των αρχείων δεν υπερβαίνουν το *PATH_MAX* (μεταβλητή ορισμένη για αυτό το λόγο στο *linux/limits.h*).
- *Για τα 2 τελευταία bash scripts*: Τα μονοπάτια των αρχείων δεν περιέχουν χαρακτήρες ':'. Αν και αυτό δεν είναι απαγορευτικό για άλλα κομμάτια του προγράμματος, η ύπαρξη αυτών στα μονοπάτια των αρχείων ενδέχεται να οδηγήσει σε εκτύπωση λάθος αποτελεσμάτων από τα συγκεκριμένα scripts καθώς ο χαρακτήρας ':' χρησιμοποιείται στα logfiles ως delimeter.

Πρόσθετες Λειτουργίες:

- Εντολή *"/pids"*: Προβολή μιας λίστας με τα process ids όλων των ενεργών διεργασιών (χρήσιμο για πειραματισμό με «σκότωμα» συγκεκριμένων διεργασιών και έλεγχο διαχείρισης των διαφόρων signals).
- Εντολή *"/help"*: Εκτυπώνει στο τερματικό τις διαθέσιμες εντολές του προγράμματος καθώς και τον τρόπο με τον οποίο αυτές συντάσσονται.
- Εντολή *"/exit -l"*: Επέκταση της εντολής *"/exit"* διαγράφοντας επίσης όλα τα log files.