Ονοματεπώνυμο: Ανδρέας – Θεολόγος Σπανόπουλος
ΑΜ: 1115201700146

ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ

- Σχεδόν όλες οι συναρτήσεις και διεργασίες έχουν παραμετροποιηθεί, δηλαδή είναι generic.
- Όλα τα προγράμματα καλούνται με χρήση fork() και execvp().
- Οι χρόνοι που μετρούνται για τους coaches και sorters είναι ουσιαστικά το cpu_time που μας δίνεται στο παράρτημα στην εκφώνηση.
- Τα αρχεία κώδικα είναι:

    1. coordinator.c
    2. coach.c
    3. sorter.c
    4. quicksort.c
    5. heapsort.c
    6. handler.c

- Τα αρχεία επικεφαλίδας είναι:

    1. records.h
    2. handler.h

- Το output είναι όπως ζητείται στην εκφώνηση.
- Χρησιμοποιώ κανονικά pipes (και όχι named pipes).
- Έχω αρκετά σχόλια μέσα στο κώδικα έτσι ώστε να μπορέσω στο README να δώσω μια σύντομη περιγραφή χωρίς να μακρηγορήσω.

coordinator.c

Όπως αναφέρεται στην εκφώνηση, ο coordinator είναι η “άγκυρα” του προγράμματος. Σε αυτόν έρχονται οι πληροφορίες, και αυτός τις εκτυπώνει. Η δομή του ουσιαστικά είναι:

1. Ξεκινάμε ελέγχοντας την ορθότητα των παραμέτρων γραμμής εντολής. Αν υπάρχει κάποιο σφάλμα, το πρόγραμμα ολοκληρώνεται. Αλλιώς, οι τιμές των παραμέτρων μεταφέρονται σε μεταβλητές (inputfile, attributes_to_sort, sort_function, number_of_records, number_of_attributes) και από εκεί τις περνάμε αργότερα στις διεργασίες – παιδιά (sorters) μέσω του execvp().

2. Στη συνέχεια δεσμεύουμε χώρο και δημιουργούμε τα pipes που θα χρησιμοποιούν οι θυγατρικές διεργασίες για να επικοινωνούν με τις γονεϊκές.
3. Έπειτα δημιουργούμε τις θυγατρικές εργασίες με χρήση fork() και εκτελούμε το πρόγραμμα coach με τις αντίστοιχες παραμέτρους μέσω χρήση execvp().
4. Βγαίνοντας από το for loop που δημιουργεί νέες διεργασίες, περιμένουμε να τερματίσουν οι θυγατρικές διεργασίες με χρήση του system call wait().
5. Διαβάζουμε τις πληροφορίες από τα pipes, και τα εκτυπώνουμε στην οθόνη.
6. Τέλος, αποδεσμεύουμε ότι μνήμη έχουμε δεσμεύσει.

Να σημειωθεί πως οι περισσότερες διαδικασίες (δημιουργία pipes, ανάγνωση πληροφοριών στα pipes, εύρεση στοιχείων, δέμευση-αποδέμευση μνήμης, κλπ..) έχουν υλοποιηθεί με generic συναρτήσεις οι οποίες έχουν οριστεί κάτω από τη main().

coach.c

Η διεργασία coach μπορεί να κληθεί μόνο από τον coordinator, με συγκεκριμένες παραμέτρους. Η δομή του του αρχείου έχει ως εξής:

1. Αρχικά ορίζουμε τη global μεταβλητή signals_arrived, η οποία αποθηκεύει τον αριθμό των σημάτων SIGUSR2 που έχουν φτάσει στον coach. Έπειτα στη main() και μέσω των αρχείων handler.c και handler.h “ορίζουμε” μια ρουτίνα διαχείρισης σήματος.
2. Στην αρχή της main() αρχικοποιούμε τις μεταβλητές του προγράμματος. Μετά δημιουργούμε τα pipes μέσω των οποίων οι sorters θα στείλουν τις ταξινομημένες εγγραφές στον coach, και ορίζουμε δύο πίνακες ακεραίων left[] και right[], η οποίοι περιέχουν τα “σύνορα” του αρχείου εγγραφών στα οποία ο κάθε sorter θα εφαρμόσει την ταξινόμηση.
3. Έπειτα δημιουργούμε θυγατρικές διεργασίες (sorters) πάλι με χρήση fork() και execvp().
4. Στη συνέχεια, όντας στην γονεϊκή διεργασία (coach), δημιουργούμε πίνακες στους οποίους θα περαστούν οι εγγραφές από τα pipes. Αφού διαβάσουμε όλες τις εγγραφές, τις κάνουμε merge στο outputfile που δημιουργούμε.
5. Στη συνέχεια αποδεσμεύουμε τη μνήμη που έχουμε δεσμεύσει, περιμένουμε να ολοκληρωθούν οι διεργασίες – sorters με χρήση του system call wait().
6. Τέλος, υπολογίζουμε τον χρόνο εκτέλεσης του coach και τον περνάμε στο αντίστοιχο pipe.

Να σημειωθεί πως και εδώ όλες οι διαδικασίες είναι generic, δηλαδή έχουν υλοποιηθεί ως συναρτήσεις που καλούνται με παραμέτρους. Μόνο η συνάρτηση που βρίσκει τα σύνορα είναι hard – coded.

sorter.c

Σύμφωνα με το question @76 στο Piazza, ο sorter απλά θα αντικατασταθεί από μια ρουτίνα ταξινόμησης (heapsort ή quicksort), με ένα exec(). Έχω φτιάξει έναν sorter, ο οποίος ανάλογα με τα ορίσματα που του περνιούνται, εκτελεί μία εκ των ταξινομήσεων.

quicksort.c & heapsort.c

Δεν υπάρχουν πολλά να εξηγήσουμε εδώ. Οι αλγόριθμοι ταξινόμησης έχουν εμπνευστεί από το βιβλίο Introduction to Algorithms των Cormen, Rivest κλπ. Τα προγράμματα παίρνουν ως ορίσματα γραμμής εντολών το όνομα του αρχείου με τος εγγραφές, τα σύνορα left και right του αρχείου που πρέπει να ταξινομήσουν, και το pipe_fd για να περάσουν τις ταξινομημένες εγγραφές μόλις εκτελεστεί η αντίστοιχη συνάρτηση ταξινόμησης. Όταν γίνει αυτό, με χρήση του system call write() περνάμε τα ταξινομημένα records στα pipes σε batches (στο αρχείο records.h υπάρχει η μακροεντολή #define BATCH 100, αν θέλετε μπορείτε να την αλλάξετε σε 50 πχ).
Τέλος, μετράμε τον χρόνο εκτέλεσης της ταξινόμησης και τον περνάμε και αυτόν σε ένα pipe(), και στέλνουμε σήμα SIGUSR2 στον coach.

record.h

Ένα header αρχείο στο οποίο έχουμε ορίσει το struct Record, και κάποιες βοηθητικές συναρτήσεις επί της δομής αυτής.

handler.c & handler.h

Αρχεία που βοηθάνε στη διαχείριση των σημάτων SIGUSR2 που στέλνουν οι sorters στους coaches.

ΧΡΗΣΙΜΕΣ ΠΛΗΡΟΦΟΡΙΕΣ

- Ο κώδικας έχει αναπτυχθεί στον κειμενογράφο Atom. Κατά συνέπεια, θα πρότεινα να χρησιμοποιηθεί αυτός για την ανάγνωσή του.
- Όλες οι απαιτήσεις της εκφώνησης έχουν καλυφθεί.
- Το testing του κώδικα και η εκτέλεση του έχει γίνει στον προσωπικό μου υπολογιστή, ο οποίος έχει την έκδοση: “gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0” του gcc. Καλού κακού θα το τρέξω και στα μηχανήματα της σχολής.
- Έχει υλοποιηθεί και ένα Makefile, για την εύκολη μεταγλώττιση του προγράμματος και τη διαγραφή αντικείμενων, εκτελέσιμων και output αρχείων. Παρακάτω θα εξηγήσω με λεπτομέρεια τη λειτουργία του.
- Ότι μνήμη δεσμεύεται, αποδεσμεύεται στο τέλος του προγράμματος.
- Οι πολυπλοκότητες των αλγορίθμων ταξινόμησης είναι θ(nlogn) στη μέση και στη βέλτιστη περίπτωση. Στη χειρότερη περίπτωση όμως, η ταχυταξινόμηση quicksort είναι θ(n^2), ενώ η ταξινόμηση σωρού heapsort παραμένει θ(nlogn).
- Για τη λειτουργία των pipes, χρησιμοποίησα το εξής tutorial που βρίσκεται στο link https://www.geeksforgeeks.org/pipe-system-call/.

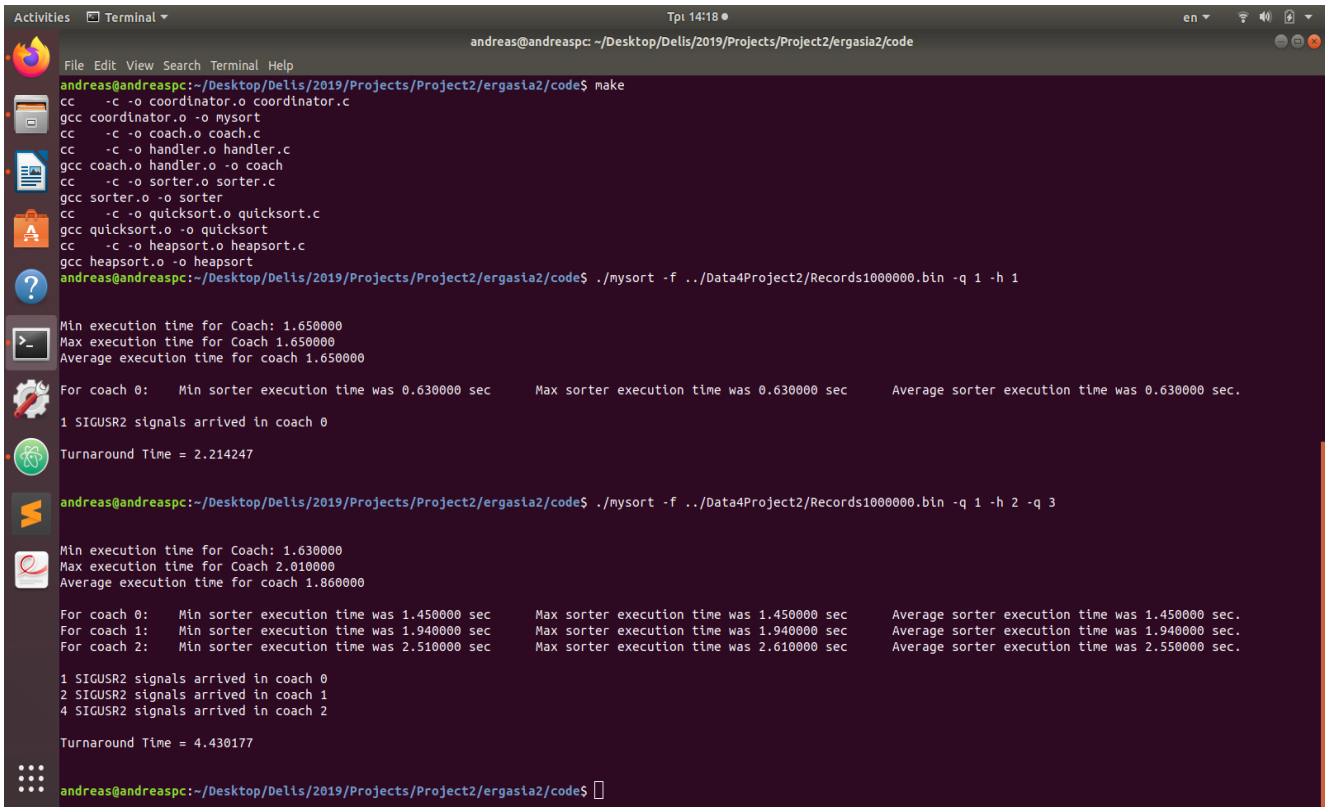
Επεξήγηση makefile

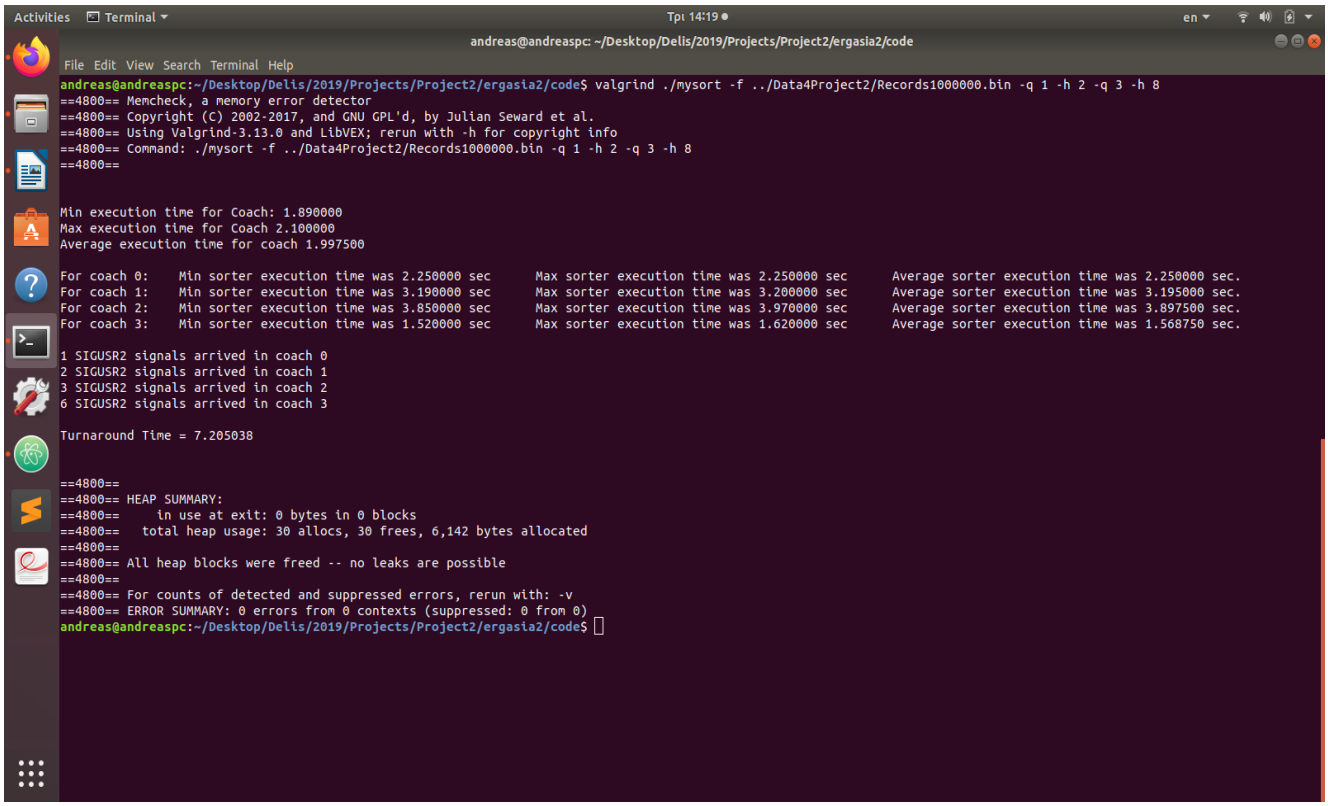
- Με την εντολή “make”, μεταγλωττίζονται όλα τα προγράμματα.
- Με την εντολή “make clean”, σβήνονται όλα τα πηγαία και τα εκτελέσιμα αρχεία.
- Με την εντολή “make rm_files”, σβήνονται τα αρχεία εξόδου που παράγει το πρόγραμμα (inputfile.*).

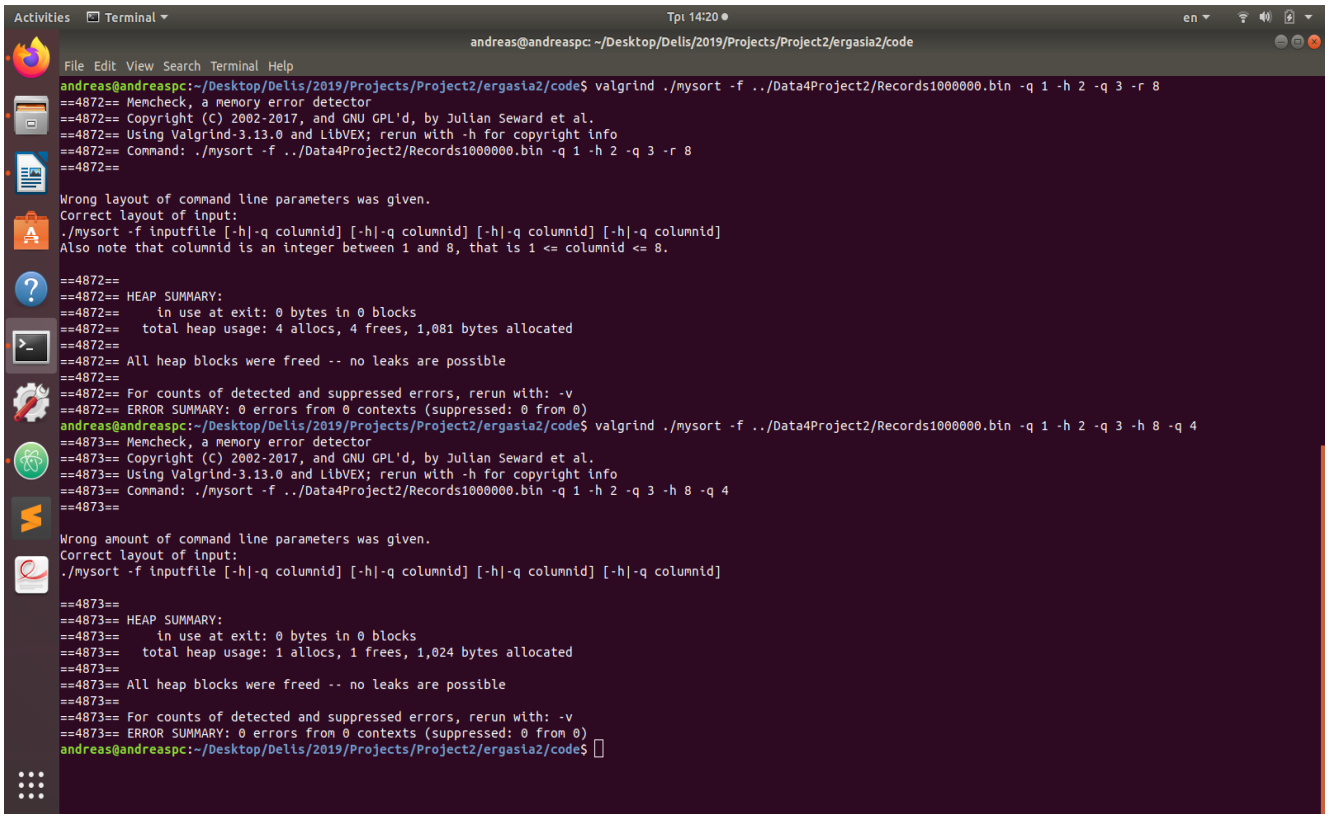
Πως να τρέξετε το πρόγραμμα

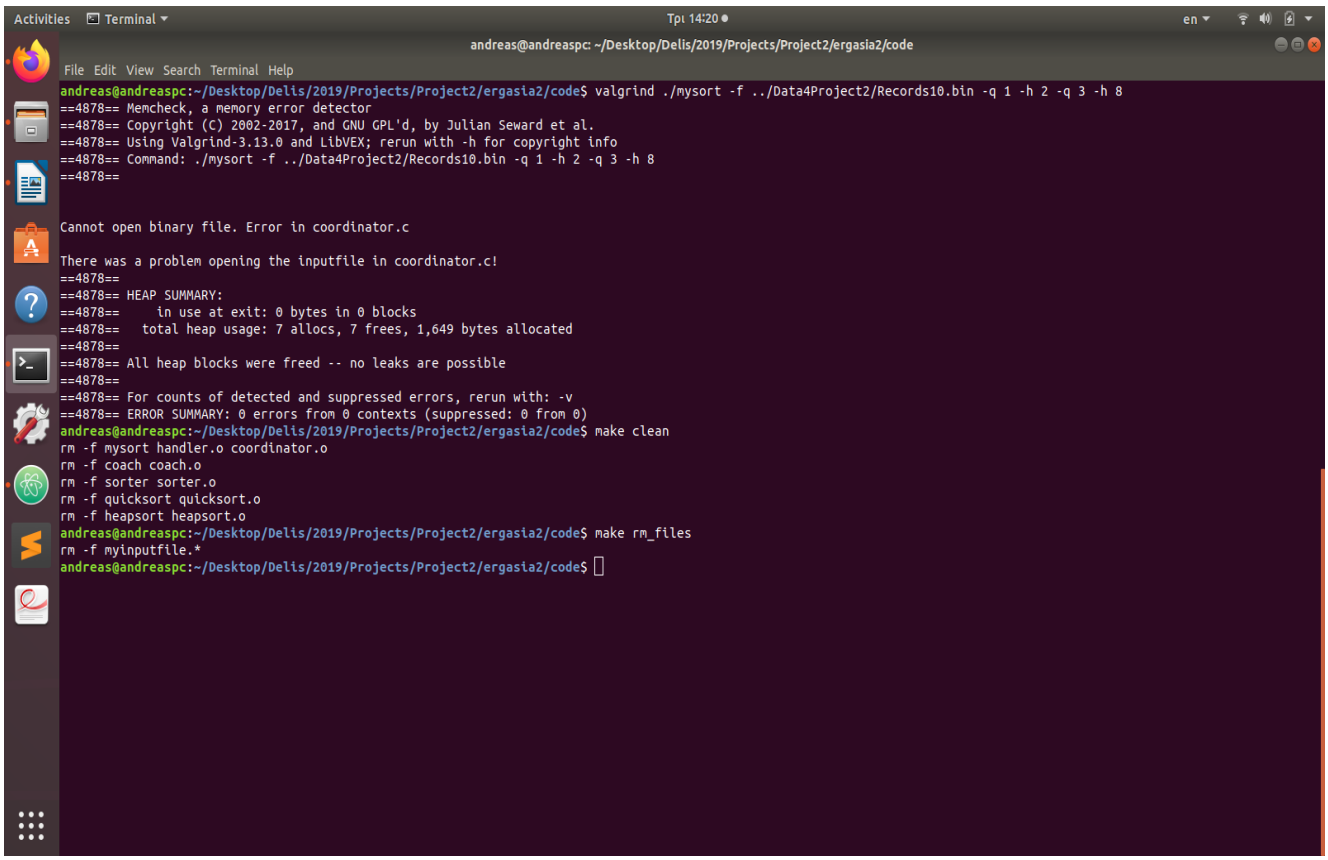
- Ξεκινάτε μεταγλωττίζοντας τα αρχεία με την εντολή “make” στο terminal.
- Με την εντολή αυτή, δημιουργείται το εκτελέσιμο αρχείο mysort.
- Για την εκτέλεση, απλά δίνετε την εντολή “./mysort -f inputfile [-h|-q columnid] [-h|-q columnid] [- h|-q columnid] [-h|-q columnid]” στο terminal. Η σειρά των παραμέτρων γραμμής εντολής δεν έχει σημασία.
- Μπορείτε να δώσετε από 0 εώς 4 επιλογές για το sorting αλγόριθμο (-h ή -q) και το πεδίο ταξινόμησης (columnid). Αν δεν δοθεί καμία επιλογή, τότε από default κάνουμε ταξινόμηση στο 1ο πεδίο με τον αλγόριθμο quicksort (με χρήση του coach 0).
- Μπορείτε να χρησιμοποιήσετε και valgrind για να τσεκάρετε πως δεν υπάρχουν memory leaks, απλώς προσθέτοντας τη λέξη valgrind μπροστά από την προηγούμενη εντολή εκτέλεσης.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ






Activities
Terminal
Τρι 14:20
en
andreas@andreaspc: ~/Desktop/Delis/2019/Projects/Project2/ergasia2/code
File Edit View Search Terminal Help
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$ valgrind ./mysort -f ../Data4Project2/Records1000000.bin -q 1 -h 2 -q 3 -r 8
==4872== Memcheck, a memory error detector
==4872== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4872== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4872== Command: ./mysort -f ../Data4Project2/Records1000000.bin -q 1 -h 2 -q 3 -r 8
==4872==
Wrong layout of command line parameters was given.
Correct layout of input:
./mysort -f inputfile [-h|-q columnid] [-h|-q columnid] [-h|-q columnid] [-h|-q columnid]
Also note that columnid is an integer between 1 and 8, that is 1 <= columnid <= 8.
==4872==
==4872== HEAP SUMMARY:
==4872== in use at exit: 0 bytes in 0 blocks
==4872== total heap usage: 4 allocs, 4 frees, 1,081 bytes allocated
==4872==
==4872== All heap blocks were freed -- no leaks are possible
==4872==
==4872== For counts of detected and suppressed errors, rerun with: -v
==4872== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$ valgrind ./mysort -f ../Data4Project2/Records1000000.bin -q 1 -h 2 -q 3 -h 8 -q 4
==4873== Memcheck, a memory error detector
==4873== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4873== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4873== Command: ./mysort -f ../Data4Project2/Records1000000.bin -q 1 -h 2 -q 3 -h 8 -q 4
==4873==
Wrong amount of command line parameters was given.
Correct layout of input:
./mysort -f inputfile [-h|-q columnid] [-h|-q columnid] [-h|-q columnid] [-h|-q columnid]
==4873==
==4873== HEAP SUMMARY:
==4873== in use at exit: 0 bytes in 0 blocks
==4873== total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4873==
==4873== All heap blocks were freed -- no leaks are possible
==4873==
==4873== For counts of detected and suppressed errors, rerun with: -v
==4873== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$


Activities
Terminal
Τρι 14:20
en
andreas@andreaspc: ~/Desktop/Delis/2019/Projects/Project2/ergasia2/code
File Edit View Search Terminal Help
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$ valgrind ./mysort -f ../Data4Project2/Records10.bin -q 1 -h 2 -q 3 -h 8
==4878== Memcheck, a memory error detector
==4878== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4878== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4878== Command: ./mysort -f ../Data4Project2/Records10.bin -q 1 -h 2 -q 3 -h 8
==4878==
Cannot open binary file. Error in coordinator.c
There was a problem opening the inputfile in coordinator.c!
==4878==
==4878== HEAP SUMMARY:
==4878== in use at exit: 0 bytes in 0 blocks
==4878== total heap usage: 7 allocs, 7 frees, 1,649 bytes allocated
==4878==
==4878== All heap blocks were freed -- no leaks are possible
==4878==
==4878== For counts of detected and suppressed errors, rerun with: -v
==4878== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$ make clean
rm -f mysort handler.o coordinator.o
rm -f coach coach.o
rm -f sorter sorter.o
rm -f quicksort quicksort.o
rm -f heapsort heapsort.o
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$ make rm_files
rm -f myinputfile.*
andreas@andreaspc:~/Desktop/Delis/2019/Projects/Project2/ergasia2/code$

UPDATE

Επειδή έχει προκύψει ένα μπέρδεμα σχετικά με την ιεραρχία και τον ρόλο των sorters, σύμφωνα με τις διευκρινίσεις που μας έδωσε ο Κύριος Δελής στο μάθημα, υπάρχουν 2 περιπτώσεις (δεν κατάλαβα ποια από τις δύο θέλει..):

- Οι διεργασίες quicksort και heapsort είναι θυγατρικές της διεργασίας sorter. Δηλαδή εκτελούνται μέσω fork() και exec() από τον sorter. Ο κώδικας που έχω παραδώσει λειτουργεί με αυτό το σενάριο, καθώς στην εκφώνηση αναφέρει πως ο sorter θα στέλνει σήμα SIGUSR2 στον coach.

- Οι διεργασίες quicksort και heapsort αντικαθιστούν τη διεργασία sorter, και στέλνουν αυτές το σήμα SIGUSR2 στον coach.

Επειδή όπως προανέφερα, δεν είμαι σίγουρος ποια από τις δύο λογικές πρέπει να ακολουθηθεί, υλοποίησα και τις δύο.

Στην υλοποίηση που σας στέλνω, η default λογική είναι η 2η. Για να εκτελέσετε το πρόγραμμα βασισμένο στο 1ο bullet, θα πρέπει να αποσχολιάσετε και να σχολιάσετε τις εξής γραμμές:

sorter.c:
Σχολιάζετε τις γραμμές 72 – 104, και αποσχολιάζετε τις γραμμές 10 – 68.

quicksort.c:
Σχολιάζετε τις γραμμές: 2, 74, 151 – 152.

heapsort.c
Σχολιάζετε τις γραμμές: 2, 68, 144 – 145.

UPDATE 2

Σχετικά με το ποιους χρόνους θα μετράμε:

Αν και η απάντηση στο question @70 του Piazza είναι αρκετά ξεκάθαρη, better safe than sorry. Στην υλοποίηση που σας έχω παραδώσει τώρα, μετράω το real time εκτέλεσης για κάθε coach και κάθε sorter του. Σε περίπτωση που τελικά θέλετε το cpu time, αρκεί πολύ απλά να:

quicksort.c

Σχολιάσετε τη γραμμή 144 και να αποσχολιάσετε τη γραμμή 157.

heapsort.c

Σχολιάσετε τη γραμμή 137 και να αποσχολιάσετε τη γραμμή 140.