ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
2η Εργασία - Τμήμα: Αρτίων Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '19
Ημερομηνία Ανακοίνωσης: Παρασκευή 25 Οκτωβρίου 2019
Ημερομηνία Υποβολής: Τετάρτη 20 Νοεμβρίου 2019 Ώρα 23:59

Εισαγωγή στην Εργασία:

Στα πλαίσια της εργασίας θα πρέπει να γράψετε ένα πρόγραμμα (`mysort`) το οποίο δημιουργεί νέες διεργασίες με τη βοήθεια της κλήσης συστήματος `fork()` και θα μπορεί να φορτώσει επιλεκτικά διαφορετικά εκτελέσιμα με την χρήση κλήσεων `exec*()`. Η ιεραρχία διεργασιών που συνθέτει το πρόγραμμα σας θα έχει τη μορφή ενός γενικού δέντρου (general tree)με εσωτερικούς κόμβους και κόμβους-φύλλα.

Ο σκοπός της άσκησης είναι η εκτέλεση πολλαπλών ταξινομήσεων (sorting) σε ένα δυαδικό αρχείο εγγραφών και η σύνθεση σχετικών αποτελεσμάτων. Οι διεργασίες του δέντρου συνολικά υλοποιούν την πολλαπλές ταυτόχρονες ταξινομήσεις αρχείου εισόδου ανάλογα με τις σημαίες εκτέλεσης. Οι κόμβοι–φύλλα, ονομαζόμενοι και sorters, είναι εκτελέσιμα που θα αναπτύξετε και υλοποιούν αλγορίθμους ταξινόμησης εγγραφών. Οι sorters εν γένει εκτελούνται ταυτόχρονα, δουλεύουν με τμήματα του αρχείου εισόδου, και *προωθούν* στους γονείς τους τα αποτελέσματα των εργασιών τους. Οι εσωτερικοί κόμβοι της ιεραρχίας ή coaches συντονίζουν την εργασία των υποκείμενων sorters, συνθέτουν τα επιμέρους αποτελέσματα και παράγουν σχετικά αρχεία εξόδου.

Στα πλαίσια αυτής της άσκησης θα κάνετε τα εξής:

- Δημιουργία ιεραρχίας διεργασιών με την κλήση συστήματος `fork()`.
- Εκτέλεση διαφόρων προγραμμάτων χρήστη/συστήματος από τις διεργασίες της ιεραρχίας με κλησεις `exec*()`.
- Χρήση ενός αριθμού από κλήσεις συστήματος (π.χ. `fork()`, `exec*()`, `getpid()`, `getppid()`, `wait()`, `waitpid()`, `read()`, `write()`, `pipe()`, `mkfifo()` κ.α.).

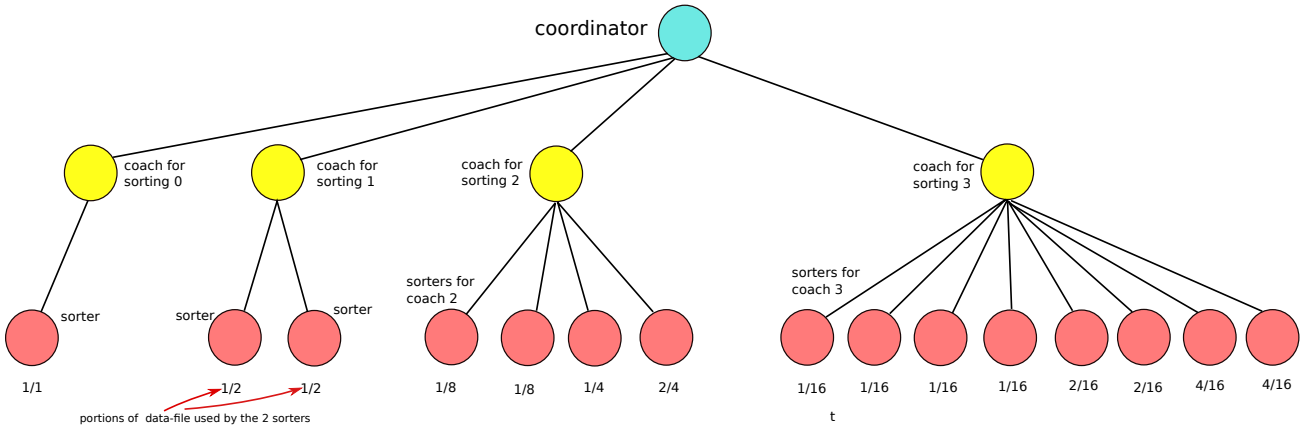
Διαδικαστικά:

Το πρόγραμμά σας θα πρέπει να γραφτεί σε *C* (ή *C++* αν θέλετε αλλά χωρίς την χρήση STL/Templates) και να τρέχει στις μηχανές `Linux workstations` του τμήματος.

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) είναι ο κ. Γιώργος Αποστολόπουλος `cs2180003+AT-di`, η κ. Αριάννα Τριανταφύλλου `cs2180017+AT-di`, και ο κ. Οδυσσέας Τρισπιώτης `odytrisp+AT-di`.
- Παρακολουθείτε την ιστοσελίδα του μαθήματος `http://www.di.uoa.gr/~ad/k22/` για επιπρόσθετες ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο URL `https://piazza.com/uoa.gr/fall2019/k22/home`
- Το πρόγραμμα σας (source code) πρέπει να αποτελείται από ***τουλάχιστον δυο*** (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία. Το πρόγραμμά σας θα πρέπει *απαραιτήτως να κάνει χρήση* separate compilation .

Η Ιεραρχία των Διεργασιών :

Το Σχήμα 1 δείχνει μια αντιπροσωπευτική ιεραρχία διεργασιών που το πρόγραμμά σας θα πρέπει να δημιουργήσει. Ο στόχος αυτής της ιεραρχίας είναι να βοηθήσει να γίνουν *πολλαπλές ταυτόχρονες ταξινομήσεις* του ιδίου αρχείου εισόδου χρησιμοποιώντας ***διαφορετικά πεδία***. Το αρχείο εισόδου δεν έχει γνωστό εκ των προτέρων μέγεθος όσον αφορά τις εγγραφές του και παρέχεται σε binary format.



Σχήμα 1: Διάταξη ιεραρχίας διεργασιών

Το δένδρο έχει συνολικά τον κοινό στόχο της ταξινόμησης εγγραφών για τον οποίο συνεργάζονται οι sorters/coaches και ο coordinator. Οι διαφορετικοί κόμβοι εκτελούν ***διαφορετικές λειτουργίες*** ανάλογα με το ρόλο που έχουν αναλάβει: οι κόμβοι sorters υλοποιούν προγράμματα που δέχονται ένα εύρος εγγραφών από το αρχείο εισόδου και το ταξινομούν είτε με το Quick–Sort ή το Heap-Sort. Οι coaches αναλαμβάνουν να δημιουργήσουν τον σωστό αριθμό από sorters στους οποίους αναθέτουν την επιμέρους ταξινόμηση που πρέπει να κάνουν, και στο τέλος συνθέτουν τα αποτελέσματα. Επίσης περνούν πληροφορίες όσον αφορά στην εκτέλεση όλων των υποκείμενων sorters στη διαδικασία coordinator. Η τελευταία κάνει την ενορχήστρωση όλων των διεργασιών και τυπώνει στατιστικά για τις διάφορες ταξινομήσεις που έχουν γίνει.

Γενικά, οι κόμβοι–διεργασίες 'παίρνουν παραμέτρους' από τους γονείς τους ώστε να κάνουν την συγκεκριμένη δουλειά που τους έχει ανατεθεί και επίσης παράγουν μερικά αποτέλεσμα/-τα και στατιστικά τα οποία τα επιστρέφουν συνήθως στην διεργασία μητέρα τους.

Λειτουργία Κόμβων:

Οι λειτουργίες των διαφόρων κόμβων (διεργασιών) έχουν ως εξής:

1. ***sorter node:*** η διαδικασία λαμβάνει από την μητέρα της το όνομα του αρχείου και το εύρος της ταξινόμησης (από ποια εγγραφή μέχρι ποια θα πρέπει ο συγκεκριμένος κόμβος να ταξινομήσει).

   Η ταξινόμηση γίνεται με 2 αυτόνομα προγράμματα που υλοποιούν το Quick-Sort και Heap–Sort. Ο κάθε sorter εκτελεί το πρόγραμμα που του υποδεικνύει ο coach του και παράγει την σχετική έξοδο.

   Τα αποτέλεσμα θα πρέπει να περάσει στην διαδικασία γονιός (coach) με την βοήθεια ενός απλού pipe ή ενός *named pipe (FIFO)*. Επίσης με τον ίδιο τρόπο μπορούν να περάσουν στο γονιό στατιστικά για τον χρόνο εκτέλεσης του κόμβου.

2. ***coach node:*** η διαδικασία λαμβάνει από την μητέρα της το όνομα του αρχείου, και τον αριθμό εγγραφών που εμπεριέχει.

   Το πρόγραμμα σας μπορεί να δημιουργήσει μέχρι και 4 coach κόμβους που ο καθένας αναλαμβάνει να υλοποιήσει ταξινόμηση σε διαφορετικό πεδίο. 4 είναι ο μέγιστος αριθμός από πεδία που μπορούν να χρησιμοποιηθούν στην ταξινόμηση.

   Ο 1ος coach δημιουργεί 2^0 sorters, ο 2ος coach δημιουργεί 2^1 sorters, ο 3ος coach, αν υπάρχει, δημιουργεί 2^2 sorters και ο 4ος coach, αν υπάρχει, 2^3 sorters.

Στο τέλος της λειτουργίας του, ένας κόμβος *coach* παραλαμβάνει τα αποτελέσματα των παιδιών του (με την βοήθεια των δύο υφιστάμενων *pipes* ή *named pipes–FIFOs*), και τα συγχωνεύει. Το νέο σύνθετο αποτέλεσμα αποθηκεύεται σε ένα καινούργιο αρχείο που έχει *ποστφιξ* το αριθμό του πεδίου πάνω στο οποίο έγινε η ταξινόμηση.

Τέλος ο coach προωθεί στο γονιό (coordinator) στατιστικά για το χρόνο εκτέλεσης των παιδιών του (πάλι με *pipe/FIFO*) και οποιαδήποτε άλλη πληροφορία κρίνεται απαραίτητη ή βοηθά στην εκτέλεση του προγράμματος.

3. ***coordinator node:*** είναι ουσιαστικά το πρόγραμμα σας που λειτουργεί σαν 'άγκυρα' του υπό δημιουργία δέντρου και επιβλέπει την συνολική ενορχήστρωση του εγχειρήματος.

   Ανάλογα με τον αριθμό των παραμέτρων που δέχεται ο coordinator δημιουργεί απο 1 εως 4 coaches στους οποίους περνούν οι κατάλληλοι παράμετροι. Για παράδειγμα αν το πρόγραμμα κάνει ταυτόχρονα 4 ταξινομήσεις, η 4η την οποία την αναλαμβάνει ο coach3 δημιουργεί 8 sorters οι οποίοι αναλαμβάνουν την διεκπεραίωση 1/16, 1/16, 1/16, 1/16, 2/16, 2/16, 4/16 και 4/16 των εγγραφών του αρχείου εισόδου (βλέπε Σχήμα 1). Ανάλογα εξελίσσονται σε αυτή την περίπτωση και οι ταξινομήσεις που υποστηρίζονται από coach0, coach1 και coach2.

   Μόλις ο coordinator συγκεντρώσει όλα τα στατιστικά, τα τυπώνει στο `tty` (δηλ. το μέσο χρόνο εκτέλεσης των παιδιών της) και ολοκληρώνει την εργασία της ιεραρχίας.

Νέες διαδικασίες στην ιεραρχία δημιουργούνται με `fork()` και όταν αυτό είναι απαραίτητο το address space των νέων αυτών διεργασιών επικαλύπτεται με `exec*()` που θα πρέπει να δέχεται τα κατάλληλα ορίσματα όπως το υπό αναζήτηση αρχείο το εύρος αναζήτησης, το βάθος του δυαδικού δένδρου που πρέπει να δημιουργηθεί και οτιδήποτε άλλο είναι απαραίτητο. Για την επικοινωνία των διεργασιών με γονείς/παιδιά θα πρέπει να χρησιμοποιήσετε `pipe()`, `mkfifo()`, κλπ.

Τέλος, όταν οι sorters τερματίζουν στέλνουν και ένα σήμα *SIGUSR2* στο coach που τους επιβλέπει. Ο τελευταίος μαζί με τα στατιστικά που τελικά παράγει επίσης παρέχει και τον αριθμό στο σημάτων *SIGUSR2* που έχει λάβει από τις διαδικασίες- κόμβους στο επίπεδο φύλλων.

Μορφοποίηση Δεδομένων στο Αρχείο Αναζήτησης:

Το δυαδικό αρχείο αναζήτησής αποτελείται από εγγραφές που έχουν την εξής μορφοποίηση:

- Αριθμός Μητρώου τύπου `long`
- Όνομα τύπου `char[20]`
- Επίθετο τύπου `char[20]`
- Οδός Κατοικίας τύπου `char[20]`
- Αριθμός Κατοικίας τύπου `int`
- Όνομα Πόλης τύπου `char[20]`
- Ταχυδρομικός Τομέας τύπου `char[6]`
- Μισθός τύπου `float`

Μορφοποίηση Εξόδου:

Το αποτέλεσμα μιας ταξινόμησης θα πρέπει μαζί με αρχεία αποτελέσματος που παράγονται σε ASCII μορφή να παρέχει και τα εξής στατιστικά:

- Min, Max και μέσος όρος χρόνου εκτέλεσης για τις διαδικασίες τύπου sorter για κάθε εμπλεκόμενο coach.
- Min, Max και μέσος όρος χρόνου εκτέλεσης για τις διαδικασίες τύπου coach.

– Turnaround Time για την ολοκλήρωση του προγράμματος.
– Για κάθε coach θα πρέπει να δωθεί ο αριθμός των σημάτων *SIGUSR2* που έχει παραληφθεί.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω αυστηρό τρόπο στην γραμμή εντολής (`tty`):
./mysort -f inputfile -h|q columnid [-h|q columnid]
όπου

- mysort είναι το εκτελέσιμο που θα δημιουργήσετε,
- inputfile είναι το binary αρχείο εισόδου δεδομένων.
- columnid είναι η στήλη πάνω στην οποία θα γίνει η ταξινόμηση.
- Η εμφάνιση της σημαίας -q δηλώνει ότι οι κόμβοι sorters χρησιμοποιούν το πρόγραμμα Quick-Sort για να επιτύχουν την ταξινόμηση και η σημαία -h το Heap-Sort.

  Η έλλειψη σημαίας τύπου q ή h προκαλεί ταξινόμηση του αρχείου εισόδου βασισμένη στο 1ο πεδίο μόνο.

Όλες οι παραπάνω σημαίες μπορούν να εμφανίζονται με οποιαδήποτε σειρά.

Για παράδειγμα η γραμμή κλήσης:
./mysort -f myinputfile -h 1 -q 3 -h 5
θα δημιουργήσει coach0, coach1 και coach2. Ο κόμβος coach0 θα χρησιμοποιήσει το πρόγραμμα Heap-Sort στα δεδομένα της 1ης στήλης για να ταξινομήσει το αρχείο εισόδου με την βοήθεια ενός sorter και το αποτέλεσμα θα αποθηκευτεί στο αρχείο με όνομα myinputfile.1. Ο κόμβος coach1 θα χρησιμοποιήσει το πρόγραμμα Quick-Sort στα δεδομένα της 3ης στήλης για να ταξινομήσει το αρχείο εισόδου με την βοήθεια 2 sorter με καθέναν να ασχολείστε με το μισό αρχείο. Το αποτέλεσμα θα αποθηκευτεί στο αρχείο με όνομα myinputfile.3. Ο κόμβος coach2 θα χρησιμοποιήσει το πρόγραμμα Heap-Sort στα δεδομένα της 5ης στήλης για να ταξινομήσει το αρχείο εισόδου με την βοήθεια 4 sorters ταξινομούν το 1/8, 1/8, 1/4 και τα 2/4 των εγγραφών του αρχείου εισόδου. Το αποτέλεσμα θα αποθηκευτεί στο αρχείο με όνομα myinputfile.5.

Η έξοδος του προγράμματος σας θα πρέπει να γίνεται στο `tty` εκτέλεσης του προγράμματος.

Χαρακτηριστικά του Προγράμματος που Πρέπει να Γράψετε:

1. Δεν μπορείτε να κάνετε pre-allocate οποιοδήποτε χώρο αφού η δομή(-ές) θα πρέπει να μπορεί(-ουν) να μεγαλώσει(-ουν) χωρίς ουσιαστικά κανέναν περιορισμό όσον αφορά στον αριθμό των εγγραφών που μπορούν να αποθηκεύσουν. Η χρήση στατικών πινάκων/δομών που δεσμεύονται στην διάρκεια της συμβολομετάφρασης του προγράμματος σας δεν είναι αποδεκτές επιλογές.
2. Για την δομή που θα υιοθετήσετε θα πρέπει να μπορείτε να εξηγήσετε (και να δικαιολογήσετε στην αναφορά σας) την πολυπλοκότητα που οι τεχνικές που υιοθετείτε παρουσιάζουν.
3. Πριν να τερματίσει η εκτέλεση της εφαρμογής, το περιεχόμενο των δομών απελευθερώνεται με σταδιακό και ελεγχόμενο τρόπο.
4. Αν πιθανώς κάποια κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει *να δώσετε αναφορά στη εν λόγω πηγή* είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.
5. Έχετε πλήρη ελευθερία να επιλέξετε το τρόπο με τον οποίο τελικά θα υλοποιήσετε βοηθητικές δομές.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (2-3 σελίδες σε ASCII κειμένου είναι αρκετές).

2. Οπωσδήποτε ένα `Makefile` (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγράμματος σας). Πιο πολλές λεπτομέρειες για το (`Makefile`) και πως αυτό δημιουργείται δίνονται στην ιστοσελίδα του μαθήματος.
3. Ένα `tar-file` με όλη σας την δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας την δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι **ατομικές**.
2. Το πρόγραμμα σας θα πρέπει να τρέχει στα `Linux` συστήματα του τμήματος αλλιώς **δεν μπορεί να βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.
4. Το παραπάνω ισχύει αν διαπιστωθεί *έστω και μερική άγνοια* του κώδικα που έχετε υποβάλει *ή άπλα υπάρχει υποψία* ότι ο κώδικας είναι προϊόν συναλλαγής με τρίτο/-α άτομο/α.
5. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 5% του βαθμού.
6. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

ΠΑΡΑΡΤΗΜΑ: Μέτρηση Χρόνου στο Linux

```
#include <stdio.h>      /* printf() */
#include <sys/times.h>  /* times() */
#include <unistd.h>     /* sysconf() */

int main( void ) {
    double t1, t2, cpu_time;
    struct tms tb1, tb2;
    double ticspersec;
    int    i, sum = 0;

    ticspersec = (double) sysconf(_SC_CLK_TCK);

    t1 = (double) times(&tb1);

    for (i = 0; i < 100000000; i++)
              sum += i;

    t2 = (double) times(&tb2);
    cpu_time = (double) ((tb2.tms_utime + tb2.tms_stime) -
                         (tb1.tms_utime + tb1.tms_stime));

    printf("Run time was %lf sec (REAL time) although \
            we used the CPU for %lf sec (CPU time).\n",
            (t2 - t1) / ticspersec, cpu_time / ticspersec);
}
```