ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
2η Εργασία - Τμήμα: Αρτίων Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '23
Ημερομηνία Ανακοίνωσης: Παρασκευή 3 Νοεμβρίου 2023
Ημερομηνία Υποβολής: Πέμπτη 23 Νοεμβρίου 2023 Ώρα 23:55

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με κλήσεις συστήματος που δημιουργούν νέες διεργασίες, διαχειρίζονται υπάρχουσες και βοηθούν στο συντονισμό και την ολοκλήρωση διεργασιών που όλες μαζί λειτουργούν ταυτόχρονα σε μια ιεραρχία. Το πρόγραμμα σας με όνομα `mysort`, δημιουργεί μία τέτοια ιεραρχία της οποίας οι κόμβοι συνολικά ταξινομούν εγγραφές που βρίσκονται σε ένα 'δυαδικό αρχείο εγγραφών'. Το αρχείο αυτό είναι η βασική παράμετρος στη γραμμή εντολής του `mysort` που δημιουργεί ένα *κ-αδικό δένδρο συνεργαζόμενων διεργασιών*.

Στην εργασία θα πρέπει:

1. να δημιουργήσετε την αναφερόμενη ιεραρχία διεργασιών με την κλήση `fork()`,
2. κόμβοι σε διαφορετικά επίπεδα της ιεραρχίας να εκτελούν πιθανώς *διαφορετικά* και *ανεξάρτητα εκτελέσιμα* που επικοινωνούν μεταξύ τους με σωληνώσεις και σήματα, και
3. να χρησιμοποιήσετε διάφορες κλήσεις συστήματος για να επιτύχετε τον υπολογιστικό σας στόχο. Οι κλήσεις αυτές συμπεριλαμβάνουν τις `exec*()`, `mkfifo()`, `pipe()`, `open()`, `close()`, `read()`, `write()`, `poll()`, `wait()`, `select()`, `dup()`, `dup2()`, `kill()` και `exit()`.

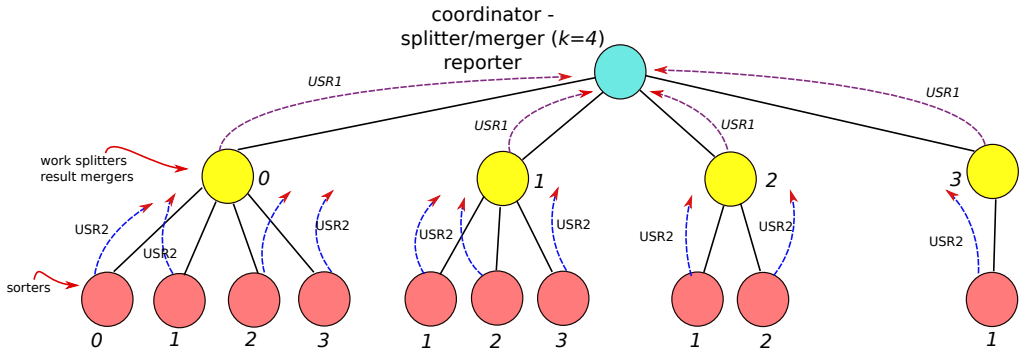
Η ιεραρχία διεργασιών που θα δημιουργήσετε, έχει σαν βασικό εκτελέσιμο το `mysort` στη ρίζα της. Το εκτελέσιμο δυναμικά δημιουργεί εσωτερικούς κόμβους και τελικά κόμβους–φύλλα.

Οι κόμβοι–φύλλα αναλαμβάνουν τη ταξινόμηση υποσυνόλων εγγραφών του αρχείου με βάση το *επίθετο* των πελατών. Όπου χρειάζεται για καλύτερο αποτέλεσμα, θα πρέπει να χρησιμοποιηθεί το όνομα αλλά και εντέλει και ο αρ. μητρώου του πελάτη. Οι εσωτερικοί κόμβοι αναλαμβάνουν τη σύνθεση όλων των ενδιάμεσων αποτελεσμάτων. Επίσης, οι εσωτερικοί κόμβοι επικοινωνούν με τα παιδιά τους αλλά και την ρίζα για την σύνθεση του αποτελέσματος μέσω σωληνώσεων (pipes ή named-pipes).

Η ταξινόμηση γίνεται από τους κόμβους φύλλα ~~που χρησιμοποιούν τουλάχιστον 2 ανεξάρτητα και διαφορετικά προγράμματα~~. Προφανώς οι διεργασίες που κάνουν την ταξινόμηση πρέπει να χρησιμοποιούν την κλήση `exec*()` για να μπορέσουν να φορτώσουν τα σωστά εκτελέσιμα στο χώρο που καταλαμβάνουν στη κυρίως μνήμη.

Διαδικαστικά:

- Το πρόγραμμα σας θα πρέπει να γραφτεί σε `C` (ή `C++` αν θέλετε αλλά χωρίς τη χρήση `STL/Templates`) και να τρέχει στα Linux *workstations* του τμήματος.
- Ο πηγαίος κώδικας σας (source code) πρέπει να αποτελείται από ***τουλάχιστον δυο*** (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία και θα πρέπει *απαραιτήτως να γίνεται χρήση* separate compilation .
- Παρακολουθείτε την ιστοσελίδα του μαθήματος `https://www.alexdelis.eu/k22/` για επιπρόσθετες ανακοινώσεις.
- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) είναι ο Δρ. Σαράντης Πασκαλής `paskalis+AT-di`, η κ. Άννα Καββαδά `ankavvada+AT-di`, και ο κ. Δημήτρης Ροντογιάννης `dronto+AT-di`.



Σχήμα 1: Παράδειγμα ιεραρχίας διεργασιών για την εφαρμογή `mysort` με k=4.

Σύνθεση Ιεραρχίας για το `mysort`:

Το Σχήμα 1 δείχνει τη σύνθεση της ιεραρχίας που λύνει το πρόβλημα της ταξινόμησης εγγραφών που επιθυμούμε.

Το βάθος της ιεραρχίας παραμένει πάντα σταθερό στο 2. Η ρίζα του δένδρου έχει k εσωτερικούς κόμβους και κάθε τέτοιος κόμβος έχει ένα μεταβλητό αριθμό από παιδιά (κόμβοι φύλλα) των οποίων ο αριθμός μεταβάλλεται από k έως και 1. Έτσι στο Σχήμα 1, ή ρίζα του δένδρου έχει δημιουργήσει 4 εσωτερικούς κόμβους (k=4) και οι 4 εσωτερικοί κόμβοι έχουν δημιουργήσει με την σειρά τους αντίστοιχα 4, 3, 2, και 1 παιδί(-ιά).

Κάνοντας χρήση ~~2 διαφορετικών~~ αλγορίθμων της επιλογής σας για ταξινόμηση, οι κόμβοι φύλλα ή sorters αναλαμβάνουν να δουλέψουν με συγκεκριμένο αριθμό από εγγραφές από το παρεχόμενο αρχείο εισόδου. Οι ~~2 αυτές~~ επιλογές θα είναι δικές σας υλοποιήσεις (δηλ. αυτόνομα προγράμματα) τα οποία μπορούν να φορτωθούν στους sorters με `exec*()`. ~~Διατρέχοντας τους~~ sorters ~~του Σχήματος 1 από τα αριστερά προς τα δεξιά, οι κόμβοι αυτοί χρησιμοποιούν εναλλακτικά τις 2 υλοποιήσεις σας.~~

Οι ρόλοι των κόμβων της ιεραρχίας είναι οι εξής:

- ***coordinator-splitter/merger-reporter:*** η αρχική αυτή διεργασία λειτουργεί σαν 'άγκυρα' του κ-αδικού δέντρου και κάνει τη συνολική διαχείριση του εγχειρήματος.

  Σε πρώτη φάση δημιουργεί k εσωτερικούς κόμβους ή splitters/mergers. Σε κάθε έναν από αυτούς αναθέτει την ταξινόμηση $1/k$ εγγραφών από το αρχείου εισόδου και αναμένει να συλλέξει την εργασία που ο καθένας από αυτούς τελικά θα παράξει. Ύστερα συνθέτει τα επιμέρους ταξινομημένα αποτελέσματα ώστε να προκύψει το τελικό σύνολο εγγραφών, το οποίο εμφανίζει στο `tty`.
- ***splitter/merger:*** κάθε τέτοιος κόμβος αναλαμβάνει να δημιουργήσει ένα μεταβλητό αριθμό από k μέχρι 1 παιδιά ανάλογά με τη θέση που βρίσκεται στην ιεραρχία, και σε καθένα από αυτά τα παιδιά του, αναθέτει την ταξινόμηση μερίδας των εγγραφών.

  Όταν οι υποκείμενοι κόμβοι φύλλα παράγουν τα επί μέρους ταξινομημένα κομμάτια του αρχείου εισόδου που τους έχουν ανατεθεί, αποστέλλουν τα αποτελέσματα τους στη μητέρα τους η οποία τα συνθέτει σε ένα σύνολο και τα προωθεί με τη σειρά της στον κόμβο ρίζα.

  Καθώς ο κάθε ***splitter/merger*** ολοκληρώνει το έργο του στέλνει ένα `USR1` στη ρίζα υποδηλώνοντας το τέλος της εργασίας του.
- ***sorter:*** κάθε τέτοιος κόμβος αναλαμβάνει να ταξινομήσει ένα σύνολο από εγγραφές από το αρχείο εισόδου και όταν αυτό ολοκληρωθεί, να αποστείλει τις ταξινομημένες πια εγγραφές στην μητέρα του. Αν υπάρχουν εγγραφές με το ίδιο επίθετο, η ταξινόμηση τότε θα πρέπει να βασίζεται και στο πρώτο όνομα. Αν τέλος υπάρχουν πολλαπλές εγγραφές με το ίδιο επίθετο και όνομα τότε η ταξινόμηση γίνεται χρησιμοποιώντας και τον αριθμό μητρώου του πελάτη.

  ~~Πριν ολοκληρώσει ο κάθε sorter, στέλνει ένα~~ `USR2` ~~σήμα στη ρίζα δηλώνοντάς έτσι την ολοκλήρωση της~~

δουλείας του.

Οι κόμβοι-φύλλα θα πρέπει επίσης να χρονομετρούν το συνολικό χρόνο που τους πήρε για να υπολογίσουν τα αποτελέσματα τους. Ο συνολικός χρόνος εκτέλεσης για κάθε διαδικασία sorter θα πρέπει επίσης να αποστέλλεται στο γονιό.

Η επικοινωνία μεταξύ των κόμβων γίνεται είτε με απλές είτε με μόνιμες σωληνώσεις (pipes/FIFO). Επομένως, αν υιοθετήσουμε ένα 4-αδικό δένδρο και το αρχείο εισόδου διαθέτει 10,000 εγγραφές, ο πρώτος sorter θα αναλάβει να ταξινομήσει τις πρώτες $10,000/16$ εγγραφές, ενώ το μόνο παιδί του ενδιάμεσου κόμβου 3 θα ταξινομήσει $10,000/4$ εγγραφές.

Το mysort ολοκληρώνει την εκτέλεση του τυπώνοντας στο `tty`

1. τις ταξινομημένες εγγραφές με βάση το επίθετο που βρίσκεται σε κάθε εγγραφή,
2. το χρόνο που χρειάστηκε για να ολοκληρώσει ο κάθε κόμβος-φύλλο την εργασία του.
3. τον αριθμό των σημάτων `USR1` και `USR2` που έχουν παραληφθεί από την ρίζα.

Να σημειώσουμε ότι όλες οι διαδικασίες της ιεραρχίας θα πρέπει να *μπορούν να δουλεύουν ταυτόχρονα*.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω αυστηρό τρόπο στη γραμμή εντολής (`tty`):
./mysort -i DataFile -k NumofChildren -e1 sorting1 -e2 sorting2
όπου

- mysort είναι το εκτελέσιμο που θα δημιουργήσετε,Α
- DataFile είναι το αρχείο που περιέχει τις εγγραφές για ταξινόμηση.
- NumofChildren είναι ο συγκεκριμένος σταθερός αριθμός των παιδιών που η ρίζα και οι ενδιάμεσοι κόμβοι δημιουργούν για να φορτώσουν ανεξάρτητα εκτελέσιμα.
- sorting1 είναι το 1ο ανεξάρτητο πρόγραμμα για ταξινόμηση που θα χρησιμοποιηθεί.
- sorting2 είναι το 2ο ανεξάρτητο πρόγραμμα για ταξινόμηση.

Οι παραπάνω in-line παράμετροι (και αντίστοιχες σημαίες) είναι υποχρεωτικές όσον αφορά την κλήση τους στην γραμμή εντολής. Οι σημαίες μπορούν να εμφανίζονται με οποιαδήποτε σειρά. Η ακριβής αναμενόμενη *μορφή εξόδου* για το πρόγραμμα δίνεται με *λεπτομέρεια* στη σελίδα του μαθήματος.

Μορφοποίηση Δεδομένων στο Αρχείο Εγγραφών:

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

- Αριθμός Μητρώου τύπου `int`
- Επίθετο τύπου `char[20]`
- Όνομα τύπου `char[20]`
- Ταχυδρομικός Τομέας τύπου `char[6]`

Χαρακτηριστικά του Προγράμματος που Πρέπει να Γράψετε:

1. Δεν μπορείτε να κάνετε pre-allocate με στατικό τρόπο οποιοδήποτε χώρο αφού 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 στη διάρκεια της συμβολομετάφρασης του προγράμματος σας *δεν είναι αποδεκτές επιλογές*.
2. Οποιαδήποτε σχεδιαστική επιλογή που θα υιοθετήσετε, θα πρέπει να τη δικαιολογήσετε στην αναφορά σας.
3. Έχετε πλήρη ελευθερία να επιλέξετε τον τρόπο με τον οποίο τελικά θα υλοποιήσετε τις 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 κείμενο είναι αρκετές).

2. Οπωσδήποτε ένα `Makefile` που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα compile τα πρόγραμμα σας.
3. Ένα `zip/7z` αρχείο με όλη σας τη δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας τη δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Βαθμολόγηση Προγραμματιστικής Άσκησης:

Σημεία Αξιολόγησης Άσκησης	*Ποσοστό Βαθμού (0–100)*
Ποιότητα στην Οργάνωση Κώδικα & Modularity	35%
Σωστή Εκτέλεση του mysort	20%
Αντιμετώπιση όλων Απαιτήσεων	25%
Χρήση Makefile & Separate Compilation	06%
Επαρκής/Κατανοητός Σχολιασμός Κώδικα	14%

Άλλες Σημαντικές Παρατηρήσεις:

1. Η εργασία είναι **ατομική**.
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 επίσης 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 ή με συστήματα αυτόματης παραγωγής λογισμικού ή με αποθηκευτήρια (repositories) οποιασδήποτε μορφής.
5. Αναμένουμε ότι όποια υποβάλει την εν λόγω άσκηση θα πρέπει να έχει πλήρη γνώση και δυνατότητα εξήγησης του κώδικα. Αδυναμία σε αυτό το σημείο οδηγεί σε μηδενισμό στην άσκηση.
6. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 8% του βαθμού.
7. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την υλοποίηση αυτής της άσκησης.

ΠΑΡΑΡΤΗΜΑ 1: Μέτρηση Χρόνου στο Linux

```c
#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);
}
```