Quiz 1, Operating Systems, 2021-2022

25 Οκτωβρίου - 4 Νοεμβρίου 2021

Q1

Στον κατάλογο που βρίσκεστε δημιουργήστε ένα κενό αρχείο που λέγεται `Freud quote.txt`. Στο αρχείο αυτό εισάγετε το κείμενο (όπως ακριβώς εμφανίζεται):

`"One day, in retrospect, the years of struggle will strike you as the most beautiful." S. Freud`

Στον υπάρχον κατάλογο, δημιουργείστε την ιεραρχία καταλόγου `Famous Quotes/Freud/Character Building`και μεταβείτε στον κατάλογο. Εκεί, μετακινήστε το αρχείο `Freud quote.txt` που δημιουργήσατε νωρίτερα.

Q2

Έχετε έναν φάκελο `Images`στον κατάλογό σας ο οποίος περιέχει αρχεία εικόνων σε πολλαπλούς υποφακέλους με τύπους αρχείων `.jpg` και `.png` της μορφής

`*.{jpg,png}`

π.χ.

```
Images/Vacation/Paris/Selfie01.jpg
Images/Course1501/Plot1.png
```

Επιθυμείτε να μετακινήσετε όλα τα αρχεία με μέγεθος μεγαλύτερο των 50 kilobytes σε οποιοδήποτε υποφάκελο.

Καταγράψτε την εντολή που χρειάζεστε για να το πετύχετε.

HINT: Στην πράξη υπάρχουν πολλοί τρόποι για να επιτευχθεί το παραπάνω, αλλά στην προκειμένη περίπτωση εξετάζεται η χρήση της εντολή find σε συνδυασμό με xargs.

Θεωρήστε ότι αρχικά βρίσκεστε έναν κατάλογο πάνω από το `Images`

Quiz 2, Operating Systems, 2021-2022

4 Νοεμβρίου - 25 Νοεμβρίου 2021

Γράψτε μία συνάρτηση σε περιβάλλον fish με όνομα `sort_files` και τρία ορίσματα (`file_extension`,`year`,`flag`) όπου `file_extension` είναι η κατάληξη των αρχείων, `year` είναι το έτος τροποποίησης του αρχείου και `flag` θα είναι `o` ή `n` (older vs newer). Η συνάρτηση θα βρίσκει τα αρχεία με κατάληξη `file_extension`, νεότερα ή παλαιότερα (flag `n` ή `o` αντίστοιχα) από το `year` και θα τα μεταφέρει σε φάκελο `output` στο τρέχον directory.

π.χ. Αν καλέσουμε την συνάρτηση με (`jpg`, `2021`, `n`) θα μας μεταφέρει όλα τα αρχεία `*.jpg` που τροποποιήθηκαν μετά την 01-01-2021 σε φάκελο `output` στο παρόν directory.

TIPS:

- Μην ασχοληθείτε με την οριακή περίπτωση 01-01-`year`
- Χρησιμοποιήστε την εντολή `find ./ -name` για να την εύρεση. Για παράδειγμα αν θέλαμε να βρούμε όλα τα `*.jpg` θα γράφαμε `find ./ -name "*.jpg"`
- Αναζητήστε πώς γίνεται η εύρεση με επιπρόσθετο κριτήριο την ημερομηνία τροποποίησης στην εντολή `find`.
- Για να δοκιμάσετε τον κώδικά σας, δημιουργήστε αρχεία με `touch -t`

Quiz 3, Operating Systems, 2021-2022

29 Νοεμβρίου - 14 Δεκεμβρίου 2021

Χρησιμοποιώντας τις εντολές fork() και wait() ετοιμάστε συνάρτηση η οποία:

- Δέχεται ως είσοδο `nList` λίστες με `nElem[i]` το μήκος της λίστα `i`. Οι λίστες είναι αποθηκευμένες σε δισδιάστατο πίνακα `int **numbers`. Π.χ. το 2ο στοιχείο της έβδομης λίστας είναι το `numbers[6][1]`.
- Ξεκινάει `nList` θυγατρικές διεργασίες χρησιμοποιώντας την `fork()`. Κάθε μία από αυτές είναι υπεύθυνη να βρει το μέγιστο και το ελάχιστο στοιχείο στην κάθε λίστα και να αποθηκεύσει το αποτέλεσμα σε νέο αρχείο με όνομα `minmax-<i>` όπου `<i>` ο αύξων αριθμός της λίστας (από `0` έως `nList-1`). Υπάρχει υλοποιημένη η παρακάτω συνάρτηση για την αποθήκευση μίας λίστας ακεραίων σε binary αρχείο, την οποία μπορείτε να καλέσετε στον κώδικά σας.

```
void writeBinary( char *name, int *minmax ){
  FILE *write_ptr = fopen(name,"wb");
  fwrite(minmax,sizeof(int),2,write_ptr);
  fclose( write_ptr );
}
```

Το διάνυσμα `minmax` έχει δύο στοιχεία. Το 1ο είναι η τιμή του ελάχιστου στοιχείου και το 2ο είναι η τιμή του μέγιστου στοιχείου.

- Περιμένει να ολοκληρωθούν όλες οι θυγατρικές διεργασίες πριν επιστρέψει.

Γράψτε στην φόρμα απάντησης τον κώδικα του προγράμματός σας σε `C`. Χρησιμοποιήστε τις μεταβλητές `INT_MIN` και `INT_MAX` για να βρείτε την τιμή του μικρότερου και μεγαλύτερου ακέραιου αριθμού.

Στον κώδικα σας δεν πρέπει να υπάρχουν καθόλου εντολές `printf()`. Αν το σύστημα σας επιστρέφει `Bad output from grader` σημαίνει ότι ο κώδικάς σας δεν ακολουθεί τις παραπάνω προδιαγραφές σχετικά με τις `printf()`.

Quiz 4, Operating Systems, 2021-2022

12 Δεκεμβρίου - 31 Δεκεμβρίου 2021

Σε αυτή την εργασία θα εξοικειωθούμε με τις σωληνώσεις και θα δούμε τη χρήση τους σε πραγματικά προβλήματα. Συγκεκριμένα θα υλοποιήσουμε διάφορες δυνατότητες ενός κελύφους (shell) σε περιβάλλον Unix.

Σας δίνουμε ένα σκελετό για το κέλυφος στο αρχείο `shell.c`. Ο σκελετός αποτελείται από δύο μέρη. Το πρώτο μέρος (parser) διαβάζει τις εντολές που γράφουμε στο κέλυφος. Το δεύτερο μέρος της εργασίας υλοποιεί την εκτέλεση των εντολών. Καλείστε να υλοποιήσετε το δεύτερο μέρος.

Η ανάγνωση των εντολών περιορίζεται σε πολύ απλές εντολές κελύφους όπως οι ακόλουθες:

```
ls > myfiles.txt
cat < myfiles.txt | sort | uniq | wc > wordcounts.txt
cat wordcounts.txt
rm wordcounts.txt
ls | sort | uniq | wc
rm myfiles.txt
```

Αντιγράψτε τις παραπάνω εντολές σε ένα αρχείο `basictest.sh` και προσπαθήστε να τις τρέξετε με το κέλυφος που σας δίνουμε. Πρώτα, μεταγλωττίστε το κέλυφος

```
$ gcc shell.c -o myshell
```

και δώστε το `basictest.sh` ως είσοδο

```
$ ./myshell < basictest.sh
```

Η εκτέλεση θα βγάλει σφάλμα διότι οι εντολές δεν έχουν υλοποιηθεί ακόμα. Με τις δικές σας αλλαγές θα πρέπει το αρχείο να τρέχει σωστά με το κέλυφος.

Απλές εντολές

Υλοποιήστε απλές εντολές, όπως

```
$ ls
```

Ο parser ήδη ετοιμάζει μια μεταβλητή τύπου `execcmd`. Πρέπει να υλοποιήσετε την συνθήκη `' '` στην συνάρτηση `runcmd`. Επαληθεύστε την υλοποίησή, προσπαθώντας να τρέξετε την εντολή `ls`. Δείτε για περισσότερες λεπτομέρειες το manual της εντολής `exec`, `man 3 exec`.

Ανακατεύθυνση

Υλοποιήστε εντολές ανακατεύθυνσης, όπως

```
$ echo "Unix is cool" > unix.txt
$ cat < unix.txt
```

Ο parser ήδη ετοιμάζει μια μεταβλητή τύπου `redircmd`. Πρέπει να υλοποιήσετε τις συνθήκες `'>'` και `'<'` στην συνάρτηση `runcmd`. Επαληθεύστε την υλοποίησή, προσπαθώντας να τρέξετε τις παραπάνω εντολές. Δείτε για περισσότερες λεπτομέρειες τα manual των εντολών `open` και `close`.

Διοχέτευση

Υλοποιήστε την δυνατότητα διοχέτευσης εντολών, όπως

```
$ ls | sort | uniq | wc
```

Ο parser ήδη ετοιμάζει μια μεταβλητή τύπου `pipecmd`. Πρέπει να υλοποιήσετε την συνθήκη `'|'` στην συνάρτηση `runcmd`. Επαληθεύστε την υλοποίησή, προσπαθώντας να τρέξετε τις παραπάνω εντολές. Δείτε για περισσότερες λεπτομέρειες τα manual των εντολών `fork`, `pipe`, `close` και `dup`.

Με αυτές τις αλλαγές θα πρέπει να είναι έτοιμο το πρόγραμμα να εκτελέσει σωστά το αρχικό αρχείο `test.sh` που σας δόθηκε.

Στο elearning θα ανεβάσετε μόνο την συνάρτηση `runcmd`. Όλες οι αλλαγές θα πρέπει να είναι εντός αυτής της συνάρτησης. Μπορείτε να ορίσετε βοηθητικές συναρτήσεις, όπου χρειάζεστε.

Στον κώδικα σας δεν πρέπει να υπάρχουν καθόλου εντολές `printf()`. Αν το σύστημα σας επιστρέφει `Bad output from grader` σημαίνει ότι ο κώδικάς σας δεν ακολουθεί τις παραπάνω προδιαγραφές σχετικά με τις `printf()`.

Quiz 5, Operating Systems, 2021-2022

16 Ιανουαρίου - 31 Ιανουαρίου 2022

Στην παρούσα εργασία, θα προσομοιώσουμε τους αλγόριθμους ανάθεσης μνήμης

- First Fit
- Best Fit
- Next Fit

Ορίζουμε μια δομή δεδομένων συνδεδεμένης λίστας την οποία θα χρησιμοποιήσουμε στην υλοποίηση των αλγορίθμων.

Η λίστα ορίζεται ως

```
struct memorySegment {
   uint startAddress, length;
   bool occupied;
   memorySegment * next;
}
```

Κάθε στοιχείο της λίστας περιγράφει ένα πεπερασμένο μπλοκ της μνήμης μεγέθους `length`. Το μπλοκ ξεκινάει στην διεύθυνση μνήμης `startAddress`. Αν είναι δεσμευμένο, τότε η σημαία `occupied` είναι `true`. Αλλιώς, είναι `false`. Το στοιχείο `next` δείχνει στο επόμενο στοιχείο της λίστας.

1. Υλοποίηση συναρτήσεων για τον χειρισμό της λίστας

Συμπληρώστε τη συνάρτηση που εμφανίζει τα περιεχόμενα μιας λίστας

```
void printList(memorySegment * memList) {

   memorySegment * current;

   // your code here to loop over the whole list and use
   printf("%d %d %s\n", current.startAddress, current.length,
       current.occupiedBy ? "Occupied!" : "Free");
}
```

Συμπληρώστε τη συνάρτηση που εισάγει μια νέα εγγραφή μετά από ένα στοιχείο μιας λίστας, ή φτιάχνει τη πρώτη εγγραφή σε μια λίστα.

```
void insertListItemAfter(memorySegment * current) {
   memorySegment * newItem;

   // your code here to allocate new record newitem

   if(current) // current !== NULL
     current.next = newItem;
   } else {
     // empty list; this is the first item in the list
```

```
      next = newItem;
    }
 }
```

Συμπληρώστε τη συνάρτηση που σβήνει την εγγραφή μετά από ένα στοιχείο μιας λίστας.

```
 void removeListItemAfter(memorySegment * current) {

    if(current) // current !== NULL
      // your code here to remove item current.next
    }
 }
```

2. Αναθέσεις μνήμης

Χρησιμοποιώντας τα παρακάτω πρότυπα υλοποιείστε τις συναρτήσεις ανάθεσης

```
memorySegment * assignFirst(memorySegment * memList, uint requestedMem);
memorySegment * assignBest(memorySegment * memList, uint requestedMem);
memorySegment * assignNext(memorySegment * memList, uint requestedMem);
void reclaim(memorySegment * memList, memorySegment * thisOne);
```

Οι συναρτήσεις αλλάζουν την σημαία `occupied` και επιστρέφουν δείκτη στην αντίστοιχη δομή που έγινε η ανάθεση ή `null` αν δεν υπάρχει διαθέσιμη μνήμη. Η λίστα με τα τμήματα της μνήμης είναι σταθερή και δεν αλλάζει.

3. Δυναμικές διαχωρήσεις μνήμης

Στα προηγούμενα ερωτήματα, δεν γινόταν αλλαγή στον αριθμό των τμημάτων μνήμης (δεν άλλαζε η λίστα) και μόνο άλλαζε το πεδίο `occupied`. Στις παρακάτω υλοποιήσεις θα προσπαθήσουμε να κάνουμε καλύτερη χρήση της διαθέσιμης μνήμης μεταβάλλοντας όλα τα πεδία ή και προσθέτοντας/αφαιρώντας καταχωρήσεις στη λίστα.

Με την ανάθεση ενός τμήματος, αν περισσεύει μνήμη, τότε είτε ενώνεται με το επόμενο τμήμα αν υπάρχει κι είναι ελεύθερο, ή εισάγεται νέα εγγραφή με το διαθέσιμο τμήμα μνήμης.

Το αντίστοιχο γίνεται και με την απελευθέρωση μιας μνήμης.

Χρησιμοποιώντας τα παρακάτω πρότυπα υλοποιείστε τις συναρτήσεις ανάθεσης

```
memorySegment * assignFirstDyn(memorySegment * memList, uint requestedMem);
memorySegment * assignBestDyn(memorySegment * memList, uint requestedMem);
memorySegment * assignNextDyn(memorySegment * memList, uint requestedMem);
void reclaimDyn(memorySegment * memList, memorySegment * thisOne);
```

4. Δοκιμές ορθότητας

Ο έλεγχος ορθότητας του κώδικα σας θα γίνει συγκρίνοντας την έξοδο του προγράμματος σας μετά από κατάθεση εντολών από κάποιο ίχνος με `assign` και `reclaim`.

Κάντε δοκιμές τοπικά με το παρακάτω παράδειγμα ίχνος:

```
A200
A300
R2
A400
R1
A300
R3
R4
A100
```

```
A100
A300
R6
```

Κάθε νέα γραμμή αντιστοιχεί σε ένα αίτημα. Το πρώτο γράμμα χαρακτηρίζει το αίτημα ως `assign` ή `reclaim`. Σε περίπτωση `assign`, ακολουθεί το μέγεθος σε μνήμη προς δέσμευση. Σε περίπτωση `reclain`, ακολουθεί α αριθμός του αιτήματος προς απελευθέρωση. Η αρίθμηση των αιτημάτων φαίνεται στα παρακάτω σχόλια.

```
A200            # 1
A300            # 2
R2
A400            # 3
R1
A300            # 4
R3
R4
A100            # 5
A100            # 6
A300            # 7
R6
```