Λειτουργικά Συστήματα – UNIX

Εργασία Εξαμήνου

Ονοματεπώνυμο: Σωτήριος

Επώνυμο: Δημητρακουλάκος

Αριθμός Μητρώου: Ε20040

Εξάμηνο: 4ο

Άσκηση 1 (tameio.sh):

Κώδικας:

```
#!/bin/bash
tameio1=5000
flag=0
n1=0
totaltime=0
while [ $totaltime -lt 300 -a $tameio1 -gt 0 -a $flag -ne 1 ]

do
  echo "Enter Time: "
  read time
  while [ $((time)) -le 0 ]

  do
    echo "Enter Valid Time: "
    read time
  done
  echo "Enter Value:"
  read poso
  if [ $((poso)) -lt 0 ]
    then
      posoneg=$((-1*poso))
      if [ $((posoneg)) -gt $((tameio1)) ]
        then
          flag=1
          poso2=$((posoneg-tameio1))
          echo "Not enough cash. $poso2 more euros needed to pay this request."
      fi
  fi
  if [ $((flag)) -ne 1 ]
    then
      tameio1=$(($tameio1+$poso))
  fi
  n1=$((n1+1))
  totaltime=$((totaltime+time))
done
echo "Customer Number: $n1"
echo "Total Time: $totaltime"
```

Επεξήγηση και Επεξηγηματικά Σχόλια:

Αρχικά, αρχικοποιούμε την μεταβλητή για το ταμείο με την τιμή των 5000 και την βοηθητική μεταβλητή flag, τον αριθμό των πελατών και τον συνολικό χρόνο με 0. Ανοίγουμε ένα while loop, το οποίο θα διαρκεί όσο το ταμείο έχει παραπάνω από 0 ευρώ ($tameio -gt 0), ο συνολικός χρόνος είναι λιγότερο από 300 λεπτά ($totaltime -lt 300) και η flag δεν είναι 1. Η flag σηματοδοτεί το αν πελάτης ζήτησε ποσό, μεγαλύτερο από εκείνο που διαθέτει το ταμείο, όπου σε αυτήν την περίπτωση η διαδικασία πρέπει να τερματίζεται.

Ερώτημα Α:

Αφού έχουν αρχικοποιηθεί οι τιμές και έτσι η ροή του προγράμματος μπόρεσε να μεταβεί στην πρώτη επανάληψη της while, ζητείται να εισαχθεί ο χρόνος που πρόκειται να διαθέσει ο πελάτης (γίνεται και αντίστοιχος έλεγχος εγκυρότητας με while, έτσι ώστε ο χρόνος να είναι μεγαλύτερος του 0).

Ερώτημα Β:

Όταν ο πελάτης εισάγει έγκυρο χρόνο, μετά ζητείται να εισάγει το ποσό που θέλει να καταθέσει (ή αντίστοιχα το ποσό που ζητά, αν το ποσό είναι αρνητικό). Αν το ποσό είναι αρνητικό, γίνεται έλεγχος (με if) για το αν τα λεφτά στο ταμείο επαρκούν για να ξεπληρωθεί το ζητούμενο ποσό και αν επαρκούν τότε απλά αφαιρούνται από το ταμείο, αλλιώς εμφανίζεται κατάλληλο μήνυμα με την υπολειπόμενη διαφορά για να τερματίζει μετά η διαδικασία, αφού το flag που προαναφέραμε γίνεται 1.

Μετά μέσα στη while, αν το flag δεν είναι 1, τότε το αντίστοιχο ποσό προστίθεται (και ουσιαστικά αν είναι αρνητικό γίνεται

αφαίρεση [μικρότερο από το ποσό που έχει μέσα το ταμείο]) και τέλος, ανεξαρτήτου περίπτωσης, ο αριθμός των πελατών αυξάνεται κατά 1 και ο συνολικός χρόνος αυξάνεται κατά τον χρόνο που διέθεσε ο τελευταίος πελάτης. Τέλος, όταν τερματίζει η while, λόγω κάποιων από τις προαναφερθέντες συνθήκες, εκτυπώνεται ο τελικός συνολικός αριθμός των πελατών που πέρασαν, καθώς και ο συνολικός χρόνος που διατέθηκε από όλους τους πελάτες.

Άσκηση 2 (guess.sh):

Κώδικας:

```
#!/bin/bash

sum1=0
all1=()
sum2=0
all2=()
sum3=0
all3=()
sum4=0
all4=()
sum5=0
all5=()

read -p "Directory to Search (Type 'exit' to end the sequence): " dir1
while [ "$dir1" != "exit" ]
do

  echo -e "\n\n 1. \n"
  harr1=()
  for f1 in $(find $dir1 -perm $1 -type f);
  do
    harr1+=($f1)
  done
  filenum1=${#harr1[@]}
  echo -e "Number of Files: $filenum1 \n"
  for i in ${harr1[@]}; do echo $i; done
  sum1=`expr $sum1 + $filenum1`
  all1+=(${harr1[@]})

  echo -e "\n\n 2. \n"
  harr2=()
  for f2 in $(find $dir1 -mtime $2 -type f);
  do
    harr2+=($f2)
  done
```

```
36  filenum2=${#harr2[@]}
37  echo -e "Number of Files: $filenum2 \n"
38  for i in ${harr2[@]}; do echo $i; done
39  sum2=`expr $sum2 + $filenum2`
40  all2+=(${harr2[@]})
41
42  echo -e "\n\n 3. \n"
43  harr3=()
44  for f3 in $(find $dir1 -atime $2 -type d);
45  do
46    harr3+=($f3)
47  done
48  filenum3=${#harr3[@]}
49  echo -e "Number of Directories: $filenum3 \n"
50  for i in ${harr3[@]}; do echo $i; done
51  sum3=`expr $sum3 + $filenum3`
52  all3+=(${harr3[@]})
53
54  echo -e "\n\n 4. \n"
55  harr4=()
56  for f4 in $(find $dir1 -type p,s);
57  do
58    harr4+=($f4)
59  done
60  filenum4=${#harr4[@]}
61  echo -e "Number of Pipe and Socket Type Files: $filenum4 \n"
62  for i in ${harr4[@]}; do echo $i; done
63  sum4=`expr $sum4 + $filenum4`
64  all4+=(${harr4[@]})
65
66  echo -e "\n\n 5. \n"
67  harr5=()
68  for f5 in $(find $dir1 -maxdepth 1 -type f -empty);
69  do
70    harr5+=($f5)
71  done
```

```
72      filenum5=${#harr5[@]}
73      echo -e "Number of Empty Files: $filenum5 \n"
74      for i in ${harr5[@]}; do echo $i; done
75      sum5=`expr $sum5 + $filenum5`
76      all5+=(${harr5[@]})
77
78      read -p "Directory to Search (Type 'exit' to end the sequence): " dir1
79    done
80
81    echo -e "\n\nHistory for 1. : \n\n"
82    echo -e "Total Number of Files: $sum1 \n"
83    echo -e "All Files:\n"
84    for el1 in ${all1[@]}; do echo $el1; done
85
86    echo -e "\n\nHistory for 2. : \n\n"
87    echo -e "Total Number of Files: $sum2 \n"
88    echo -e "All Files:\n"
89    for el2 in ${all2[@]}; do echo $el2; done
90
91    echo -e "\n\nHistory for 3. : \n\n"
92    echo -e "Total Number of Directories: $sum3 \n"
93    echo -e "All Directories:\n"
94    for el3 in ${all3[@]}; do echo $el3; done
95
96    echo -e "\n\nHistory for 4. : \n\n"
97    echo -e "Total Number of Pipe and Socket Type Files: $sum4 \n"
98    echo -e "All Files:\n"
99    for el4 in ${all4[@]}; do echo $el4; done
100
101   echo -e "\n\nHistory for 5. : \n\n"
102   echo -e "Total Number of Empty Files: $sum5 \n"
103   echo -e "All Files:\n"
104   for el5 in ${all5[@]}; do echo $el5; done
```

Επεξήγηση και Επεξηγηματικά Σχόλια:

Στην αρχή γίνεται αρχικοποίηση όλων των μεταβλητών (με 0), που θα κρατάνε το συνολικό αριθμό αρχείων για κάθε ερώτημα και όλων των πινάκων (ως κενοί) που θα κρατάνε όλα τα αρχεία, όλης της διαδικασίας για κάθε ερώτημα. Ύστερα, ζητείται από τον χρήστη ο κατάλογος του οποίου θέλουμε να αναζητήσουμε τα

αρχεία (σύμφωνα με τα ορίσματα που έχουν δοθεί), όπως γίνεται και στο τέλος της επερχόμενης while, κάθε φορά μέχρι να τερματίσει η διαδικασία, όταν αντί για διεύθυνση καταλόγου δοθεί η λέξη ‘exit’.

Τώρα, για κάθε ερώτημα γίνεται η εξής παρόμοια διαδικασία, με κάποιες αλλαγές στις παραμέτρους της εντολής find. Αρχικοποιείται κενός πίνακας για να μπουν μέσα μετά τα αντίστοιχα αρχεία, μετά με μια for loop κάθε αρχείο από το σύνολο αρχείων που βρίσκουμε με τη find, σύμφωνα με τα ζητούμενα του ερωτήματος, εισάγεται στον πίνακα αυτόν (η διεύθυνσή του), ύστερα μπαίνει σε αντίστοιχη μεταβλητή ο αριθμός των στοιχείων που έχει μέσα ο πίνακας (π.χ. filenum1=${#harr1[@]}), με μια for loop, εκτυπώνεται κάθε αρχείο από αυτόν τον πίνακα και τέλος προστίθεται ο αριθμός αρχείων που βρέθηκε στον συνολικό (π.χ. sum1=`expr $sum1 + $filenum1`) και τα αρχεία όλα ένα-ένα επίσης στον πίνακα με τα συνολικά. Το μόνο που διαφέρει σε κάθε ερώτημα είναι το σύνολο που βρίσκουμε με την find.

<u>Ερώτημα 1:</u>

Για να βρεθούν τα αρχεία του δέντρου του δοθέντος καταλόγου με εξουσιοδοτήσεις τον πρώτο αριθμό (όρισμα) θεωρώντας τον ως οκταδικό ισοδύναμο, βάζουμε μέσα στο σύνολο που θα προσπελάσει η for, το αποτέλεσμα της εντολής: find $dir1 -perm $1 -type f. Το -perm $1 είναι για τα permissions που δίνει το αντίστοιχο όρισμα $1 (πρώτο όρισμα) και το -type f είναι για να ψάξει κανονικά όλα τα αρχεία.

Ερώτημα 2:

Για να βρεθούν τα αρχεία του δέντρου του δοθέντος καταλόγου που άλλαξαν περιεχόμενα κατά τις ‘x’ τελευταίες μέρες, όπου ‘x’ ο δεύτερος αριθμός (όρισμα), βάζουμε μέσα στο σύνολο που θα προσπελάσει η for, το αποτέλεσμα της εντολής: find $dir1 -mtime $2 -type f. Το -mtime $2 είναι για να ψάξει σύμφωνα με το modification time (πότε έγινε η τελευταία αλλαγή), που δίνει το αντίστοιχο όρισμα $2 (δεύτερο όρισμα) και το -type f είναι για να ψάξει κανονικά όλα τα αρχεία.

Ερώτημα 3:

Για να βρεθούν οι υποκατάλογοι του δέντρου του δοθέντος καταλόγου που προσπελάστηκαν κατά τις ‘x’ τελευταίες μέρες, όπου ‘x’ ο δεύτερος αριθμός (όρισμα), βάζουμε μέσα στο σύνολο που θα προσπελάσει η for, το αποτέλεσμα της εντολής: find $dir1 -atime $2 -type d. Το -atime $2 είναι για να ψάξει σύμφωνα με το access time (πότε έγινε η τελευταία προσπέλαση/πρόσβαση), που δίνει το αντίστοιχο όρισμα $2 (δεύτερο όρισμα) και το -type d είναι για να ψάξει καταλόγους (directories).

Ερώτημα 4:

Για να βρεθούν τα αρχεία του δέντρου του δοθέντος καταλόγου που είναι τύπου pipe ή socket, βάζουμε μέσα στο σύνολο που θα προσπελάσει η for, το αποτέλεσμα της εντολής: find $dir1 -type p,s. Το -type p,s είναι για να ψάξει αποκλειστικά αρχεία τύπου pipe και socket.

Ερώτημα 5:

Για να βρεθούν τα κενά αρχεία του δοθέντος καταλόγου (όχι του δέντρου), βάζουμε μέσα στο σύνολο που θα προσπελάσει η for, το αποτέλεσμα της εντολής: find $dir1 -maxdepth 1 -type f -

empty). Το - maxdepth 1 είναι για να ψάξει μόνο τα αρχεία του δοθέντος καταλόγου, αφού το μέγιστο βάθος αναζήτησης στα directories και sub-directories είναι 1 (το ίδιο το directory) και το -type f για να ψάξει κανονικά όλα τα αρχεία, αλλά και με το -empty ψάχνει και βρίσκει μόνο κενά αρχεία.

Ερώτημα (α):

Τέλος παρουσιάζεται ένα ιστορικό για κάθε ερώτημα, όπου εκτυπώνεται ο συνολικός αριθμός των αρχείων/καταλόγων που βρέθηκαν στο καθένα.

Ερώτημα (β):

Καθώς και όλα τα αντίστοιχα ονόματα (διευθύνσεις) τους μαζί.

Άσκηση 3 (alphanum.sh):

Κώδικας:

```
#!/bin/bash

first=$(grep -n "^$2" $1 | { grep -v grep || true; })

second=$(grep -n "$2" $1)

third=$(grep -n "$2$" $1 | { grep -v grep || true; })

firstnum=$(echo "$first" | wc -l)

secondnum=$(echo "$second" | wc -l)

thirdnum=$(echo "$third" | wc -l)

echo -e "\nBeginning of Line:\n\n\n$first\n\nNumber of Lines: $firstnum"

echo -e "\n\n\n\nIn Line:\n\n\n$second\n\nNumber of Lines: $secondnum"

echo -e "\n\n\n\nEnd of Line:\n\n\n$third\n\nNumber of Lines: $thirdnum\n"

```

Επεξήγηση και Επεξηγηματικά Σχόλια:

Ερώτημα 1:

Βάζουμε στο firstnum όλες τις αριθμημένες γραμμές του δοθέντος αρχείου $1, που ξεκινάνε με το αλφαριθμητικό $2, εκχωρώντας το αποτέλεσμα της εντολής grep -n "^$2" $1 | { grep -v grep || true; } σε αυτό. Με την παράμετρο ^ μπροστά από το όρισμα, εξασφαλίζουμε να ληφθούν οι γραμμές που έχουν το $2 στην αρχή τους και με το -n αριθμούνται οι γραμμές.

Ερώτημα 2:

Βάζουμε στο secondnum όλες τις αριθμημένες γραμμές του δοθέντος αρχείου $1, που απλά περιέχουν το αλφαρηθμητικό $2, εκχωρώντας το αποτέλεσμα της εντολής grep -n "^$2" $1 σε αυτό. Με το -n αριθμούνται οι γραμμές.

Ερώτημα 3:

Βάζουμε στο thirdnum όλες τις αριθμημένες γραμμές του δοθέντος αρχείου $1, που τελειώνουν με το αλφαριθμητικό $2, εκχωρώντας το αποτέλεσμα της εντολής grep -n "2" $1 | { grep -v grep || true; } σε αυτό. Με την παράμετρο $ πίσω από το όρισμα, εξασφαλίζουμε να ληφθούν οι γραμμές που έχουν το $2 στο τέλος τους και με το -n αριθμούνται οι γραμμές.

Για κάθε ερώτημα, μετριέται ο αντίστοιχος αριθμός των γραμμών που περιέχουν οι αντίστοιχες μεταβλητές (π.χ. για το πρώτο ερώτημα, παίρνουμε την τιμή της εντολής echo "$first" | wc -l [pipline του αποτελέσματος του echo σε word count γραμμών [-l]]) και εκτυπώνεται μαζί με τις αντίστοιχες αριθμημένες γραμμές.

Άσκηση 4 (size.sh):

Κώδικας:

```bash
#!/bin/bash

if [ $1 -le 0 -o $1 -gt 14 ]
then
  echo -e "Invalid Input!\n"
else
  flag=0
  files2=()
  files0=$(ls)
  files1=(${files0// / })
  for i in "${files1[@]}"
  do
    birth=$(stat -c '%w' $i)
    split1=(${birth// / })
    time=${split1[1]}
    split2=(${time//:/ })
    hour=${split2[0]}
    if (( hour == $1 ))
    then
      flag=1
      files2+=($i)
    fi
  done
  if [ $flag -eq 0 ]
  then
    echo -e "No files created at this hour.\n"
    exit
  fi
  mkdir timefile
  for j in "${files2[@]}"; do mv $j timefile; done
fi

```

Επεξήγηση και Επεξηγηματικά Σχόλια:

Ερώτημα i):

Αρχικά, γίνεται έλεγχος του δοθέντος ορίσματος σχετικά με την τιμή του με μια if ([$1 -le 0 -o $1 -gt 14]), η οποία αν ισχύει εκτυπώνεται αντίστοιχο μήνυμα Invalid Input και το πρόγραμμα τερματίζει, αφού παραλείπεται όλο το υπόλοιπο που κομμάτι του προγράμματος που βρίσκεται στο σκέλος της else.

Ερώτημα ii):

Αν η τιμή είναι έγκυρη, ακολουθούνται τα εξής βήματα:

-Αρχικοποίηση flag με 0 και του πίνακα files2 που θα έχει μέσα τα τελικά, ζητούμενα αρχεία που θα βάλουμε στον κατάλογο timefile. Φτιάχνουμε αρχείο καταλόγου, διότι είναι το μόνο που μπορούμε να κάνουμε overwrite με οποιοδήποτε άλλο αρχείο.

-Βάζουμε στο files0 τα περιεχόμενα του καταλόγου (όλα τα αρχεία), στον οποίο βρισκόμαστε, με την εκχώρηση του αποτελέσματος της εντολής ls.

-Στον files1 βάζουμε τα περιεχόμενα του καταλόγου από τον files0, αλλά τώρα τα διαχωρίζουμε (με ${files0// / } όπου το σύμβολο μετά την δεύτερη κάθετο είναι το σύμβολο σύμφωνα με το οποίο διαχωρίζουμε), σύμφωνα με τα κενά ανάμεσα τους, έτσι ώστε να ξεχωρίζουν και να μπορούμε να επιλέξουμε το καθένα ξεχωριστά.

-Ύστερα για κάθε στοιχείο (αρχείο) στον files1 αποκομούμε πρώτα με το αποτέλεσμα της εντολής $(stat -c '%w' $i) (όπου i το

αντίστοιχο αρχείο), δηλαδή τα birth status του αρχείου που έχουν την μορφή 2022-06-30 13:57:15.000000000 -0600. Το αποτέλεσμα αυτό το χωρίζουμε με τον ίδιο τρόπο όπως πάνω (με ${birth// / } (και μπαίνει σε ένα array όπου κάθε κόμβος είναι ένα διαχωρισμένο κομμάτι), σύμφωνα με τα κενά, επιλέγουμε το δεύτερο στοιχείο (13:57:15.000000000) και τέλος χωρίζουμε και αυτό το ίδιο με τον ίδιο ακριβώς τρόπο, αφού το έχουμε βάλει σε μεταβλητή, την έχουμε χωρίσει σύμφωνα με το σύμβολο ΄:΄ (${birth//:/ }), έχουμε βάλει σε μια τελική μεταβλητή το πρώτο στοιχείο (13) και έτσι έχουμε την ώρα δημιουργίας για το αντίστοιχο αρχείο.

-Κατόπιν, ελέγχουμε αν η ώρα που αποκομίσαμε ισούται με το όρισμα $1, και αν ναι, βάζουμε το αντίστοιχο αρχείο στον πίνακα files2. Αν μπει σε αυτό το if το πρόγραμμα (δηλαδή αν υπάρχει έστω και ένα αρχείο που να δημιουργήθηκε την δοθείσα ώρα), το flag γίνεται 1, άρα δεν θα μπει στο ακριβώς επόμενο if και άρα θα φτιαχτεί το timefile. Αλλιώς αν δεν μπει ποτέ, το flag θα είναι 0 και έτσι θα εκτυπωθεί κατάλληλο μήνυμα και το πρόγραμμα θα τερματίσει.

-Τέλος, δημιουργείται το αρχείο (κατάλογος) timefile (mkdir timefile) και μεταφέρονται όλα τα αρχεία του files2 σε αυτό με την: for j in "${files2[@]}"; do mv $j timefile; Done.

Ευχαριστώ για την προσοχή και τον χρόνο σας!

***Προσωπική Σημείωση: Αυτή τη στιγμή είμαι στρατό και συγκεκριμένα ένδον υπηρεσίας με ποικίλες άλλες υποχρεώσεις και αγγαρείες, οπότε συγχωρέστε με για τυχόν απροσεξίες, ελλείψεις ή/και αστοχίες, κατά την υλοποίηση της εργασίας. Μια μικρή κατανόηση και επιείκεια θα ήταν ιδιαίτερα εκτιμητή.**

-ΣΩΤΉΡΙΟΣ ΔΗΜΗΤΡΑΚΟΥΛΆΚΟΣ – Ε20040