4ο ΕΡΓΑΣΤΗΡΙΟ

ΣΤΟΧΟΣ

Διεργασίες, ανακατεύθυνση, διασωλήνωση, φίλτρα, Εισαγωγή στα σενάρια φλοιού

Σε όλες τις παρακάτω ασκήσεις προσοχή στα copy-paste διότι το word αλλοιώνει τους πραγματικούς χαρακτήρες (", ‘, -, κτλ)

ΑΣΚΗΣΗ 1

1. Να χρησιμοποιήσετε την εντολή ps για να δείτε τις διεργασίες που εκτελούνται στο σύστημα με όλες τις δυνατές επιλογές (π.χ. ps , ps -ef, ps -al, ps -aldef κλπ).
2. Στη συνέχεια να χρησιμοποιήσετε την εντολή kill ή kill -9 σε κάποια από τις ενεργές διεργασίες. Τι παρατηρείτε;

Η εντολή kill ΔΕΝ «ΣΚΟΤΩΝΕΙ» μια διεργασία, αλλά στέλνει ένα σήμα σε αυτήν.

Τα signals που μπορεί να στείλει η εντολή kill μπορείτε να τα δείτε με:

```
asidirop@aetos:~$ /bin/kill -L
 1 HUP       2 INT       3 QUIT      4 ILL       5 TRAP      6 ABRT      7 BUS
 8 FPE       9 KILL     10 USR1     11 SEGV     12 USR2     13 PIPE     14 ALRM
15 TERM     16 STKFLT   17 CHLD     18 CONT     19 STOP     20 TSTP     21 TTIN
22 TTOU     23 URG      24 XCPU     25 XFSZ     26 VTALRM   27 PROF     28 WINCH
29 POLL     30 PWR      31 SYS
```

Ή με την εντολή man kill

Κάποια σήματα έχει νόημα να τα στείλει ο χρήστης σε μια διεργασία, ενώ κάποια άλλα όχι (χρησιμοποιούνται για εσωτερικούς σκοπούς από το λειτουργικό).

Τα περισσότερο χρησιμοποιούμενα σήματα είναι:

15 (TERM): Λέμε στην διεργασία να τερματίσει. Η διεργασία θα τερματίσει αν το θέλει.
9 (KILL): Τερματίζουμε βίαια την διεργασία. Η διεργασία θα τερματίσει είτε το θέλει είτε όχι.
19 (STOP): Λέμε στην διεργασία (ουσιαστικά στο λειτουργικό) να μπει σε STOP mode. Η διεργασία σταματά να εκτελείται, αλλά δεν τερματίζεται.
18 (CONT): Λέμε στην διεργασία που ήταν σε STOP mode να συνεχίσει να εκτελείται.

3. Υλοποιήστε το παρακάτω:
   a. Εάν έχετε εγκατεστημένο Linux ή VM Linux: Ανοίξτε 2 τερματικά (**όχι στο user**). Από το ένα τερματικό σας εκτελέστε την εντολή:
      nautilus
      Θα εκτελεστεί ο FileExplorer του gnome. Από το άλλο τερματικό, εκτελώντας την εντολή “ps ux” βρείτε το PID που αντιστοιχεί στην διεργασία του nautilus. Στείλτε στην διεργασία το σήμα STOP. (Αν το PID που βρήκατε είναι το 1234, δώστε την εντολή: kill -STOP 1234).

Τι συνέβη με την εφαρμογή? Δοκιμάστε να την χρησιμοποιήσετε, να την κάνατε resize, minimize....
Στείλτε το σήμα CONT στην ίδια διεργασία. Τι συνέβη?
Στείλτε το σήμα TERM στην ίδια διεργασία. Τι συνέβη?

b. Εάν δεν έχετε εγκατεστημένο Linux ή VM Linux , ανοίξτε δυο τερματικά στο users. Στο ένα εκτελέστε την εντολή
curl https://users.iee.ihu.gr/~asidirop/delay.php
Αυτή η εντολή κατεβάζει το παραπάνω αρχείο από το web. Όμως αυτό το αρχείο αργεί να κατέβει…. Τα περιεχόμενά του είναι αριθμοί, ένας αριθμός κάθε 1 δευτερόλεπτο. Από το άλλο τερματικό, εκτελώντας την εντολή “ps ux” βρείτε το PID που αντιστοιχεί στην διεργασία του curl. Στείλτε στην διεργασία το σήμα STOP. (Αν το PID που βρήκατε είναι το 1234, δώστε την εντολή: kill -STOP 1234). Τι συνέβη με την εφαρμογή? Στείλτε το σήμα CONT στην ίδια διεργασία. Τι συνέβη? Μπορείτε να χρησιμοποιήσετε το τερματικό στο οποίο τρέχει η curl; Στείλτε το σήμα TERM στην ίδια διεργασία. Τι συνέβη?

4. Κάντε ssh στο users. Εκτελέστε την εντολή “ps auxw”. Εντοπίστε μια διεργασία που δεν ανήκει σε εσάς και στείλτε της το σήμα KILL (π.χ.: kill -KILL 1234). Τι συνέβη?
5. Από την έξοδο της εντολής “ps auxw” μπορείτε να βγάλετε συμπεράσματα για
   a. Τις ενέργειες που κάνουν αυτή την στιγμή οι χρήστες?
   b. Τα services που εκτελούνται στο μηχάνημα?

ΑΣΚΗΣΗ 2

Μια διεργασία όταν εκτελείται από ένα τερματικό, τότε «παίρνει» τον έλεγχο του τερματικού, δηλαδή ότι πληκτρολογήσουμε θα διαβαστεί από την διεργασία (αν η διεργασία δεν θέλει να διαβάσει από το τερματικό, τότε ό,τι πατάμε παραμένει στον buffer του τερματικού μέχρι κάποιος να το διαβάσει). Το shell δεν μπορεί να διαβάσει από το τερματικό μέχρι να το αποδεσμεύσει το τερματικό η διεργασία.

Άσκηση 2: έκδοση για linux desktop

1. Ανοίξτε 1 τερματικά (**τοπικά, όχι στο users**). Από το τερματικό δώστε πάλι την εντολή "kcalc. Ο FileExplorer εκτελείται. Στο τερματικό δεν μπορούμε να δώσουμε εντολές… πληκτρολογούμε αλλά δεν γίνεται τίποτε. Πατήστε τα πλήκτρα Cntrl-Z. (συμβολίζεται με ^Z).

Το ^Z στέλνει στην τρέχουσα διεργασία το σήμα STOP.

Πλέον το παράθυρο του kcalc δεν αντιδρά τις κινήσεις μας.

Το shell έχει την δυνατότητα να χειριστεί τις διεργασίες που εκτελέσαμε μέσα από αυτό. Τις ονομάζει jobs. Ένα job είναι μια διεργασία που εκτελέστηκε από το shell και σταμάτησε (ή μπήκε στο παρασκήνιο). Πατώντας ^Z η τρέχουσα διεργασία γίνεται job.

```
asidirop@dellpc:~$ kcalc
^Z
[1]+  Stopped                 kcalc
asidirop@dellpc:~$
```

τα jobs έχουν την δική τους αρίθμηση από το shell. Στην παραπάνω περίπτωση είναι το jobs [1]. Το + μας λέει ότι είναι το τρέχον job.

2. Δώστε την εντολή (από το ίδιο τερματικό)
   kwrite &

Το & στο τέλος μιας εντολής λέει στο shell να εκτελέσει την εντολή, και να την βάλει να εκτελείται στο παρασκήνιο (δηλαδή η εντολή δεν θα μπορεί να διαβάσει από το τερματικό αν το θελήσει). Το τερματικό παραμένει στον έλεγχο του shell.

3. Δώστε την εντολή
   `jobs`

   μας δείχνει όλα τα jobs του τρέχοντος shell και το status του καθενός.

   Αν η έξοδος από την εντολή jobs είναι η:

```
asidirop@dellpc:~$ jobs
[1]+  Stopped                 kcalc
[2]-  Running                 kwrite &
```

   Δώστε την εντολή

   `bg %1`

   Δηλαδή % και τον αριθμό του job που αντιστοιχεί στο σταματημένο job.

Η εντολή bg στέλνει το σήμα CONT στην διεργασία που αντιστοιχεί, και την βάζει να εκτελείται στο

παρασκήνιο.

Εκτελέστε πάλι την εντολή jobs.

Δώστε την εντολή

```
fg %1
```

Η εντολή fg στέλνει το σήμα CONT στην διεργασία που αντιστοιχεί, και την βάζει να εκτελείται στο προσκήνιο, δηλαδή της δίνει τον έλεγχο του τερματικού.

Πατήστε τα πλήκτρα Cntrl-C (συμβολίζεται ^C).

Το ^C στέλνει στην τρέχουσα διεργασία το σήμα TERM.

Η εφαρμογή kcalc τερματίστηκε. Τώρα ως job έχουμε μόνο το kwrite.

Δώστε την εντολή:

kcharselect &

(ή «gnome-characters & »

αν δεν είναι εγκατεστημένο το προηγούμενο)

Τώρα έχουμε 2 jobs που τρέχουν.

Κλείστε το τερματικό από το οποίο τρέξατε τα προηγούμενα jobs (πατώντας στο παράθυρο το κουμπί για κλείσιμο).

Το bash, όταν τερματίζεται, τερματίζει και όλα τα jobs του (στέλνει σε όλα τα jobs το σήμα SIGHUP).

Εάν δεν θέλουμε ένα job να μην τερματιστεί με την έξοδο από το shell, τότε πρέπει να δώσουμε την εντολή disown %x (όπου x το job number).

Η εντολή disown αποδεσμεύει ένα job από το τρέχον shell. Η διεργασία όμως παραμένει να τρέχει.

Άσκηση 2: έκδοση για linux ssh

4. Ανοίξτε 1 τερματικό **στο users**. Από το τερματικό δώστε την εντολή curl -s -N https://users.iee.ihu.gr/~asidirop/delay.php > download1
   Το curl εκτελείται και κατεβάζει αποθηκεύοντας στο download1. Στο τερματικό δεν μπορούμε να δώσουμε εντολές… πληκτρολογούμε αλλά δεν γίνεται τίποτε. Πατήστε τα πλήκτρα Cntrl-Z. (συμβολίζεται με ^Z).

Το ^Z στέλνει στην τρέχουσα διεργασία το σήμα STOP.

Πλέον η curl σταμάτησε.

Το shell έχει την δυνατότητα να χειριστεί τις διεργασίες που εκτελέσαμε μέσα από αυτό. Τις ονομάζει jobs. Ένα job είναι μια διεργασία που εκτελέστηκε από το shell και σταμάτησε (ή μπήκε στο παρασκήνιο). Πατώντας ^Z η τρέχουσα διεργασία γίνεται job.

```
asidirop@users:~$ curl -s https://users.iee.ihu.gr/~asidirop/delay.php > download1
^Z
[1]+  Stopped                 curl -s -N https://users.iee.ihu.gr/~asidirop/delay.php > download1
asidirop@users:~$ jobs
[1]+  Stopped                 curl -s -N https://users.iee.ihu.gr/~asidirop/delay.php > download1
asidirop@users:~$
```

τα jobs έχουν την δική τους αρίθμηση από το shell. Στην παραπάνω περίπτωση είναι το jobs [1]. Το + μας λέει ότι είναι το τρέχον job.

5. Δώστε την εντολή (από το ίδιο τερματικό)
   curl -s -N https://users.iee.ihu.gr/~asidirop/delay.php > **download2** &

Το & στο τέλος μιας εντολής λέει στο shell να εκτελέσει την εντολή, και να την βάλει να εκτελείται στο παρασκήνιο (δηλαδή η εντολή δεν θα μπορεί να διαβάσει από το τερματικό αν το θελήσει). Το τερματικό παραμένει στον έλεγχο του shell.

6. Δώστε την εντολή
   `jobs`

   μας δείχνει όλα τα jobs του τρέχοντος shell και το status του καθενός.

   Αν η έξοδος από την εντολή jobs είναι η:

```
asidirop@dellpc:~$ jobs
[1]+  Stopped                 curl -s -N https://users.iee.ihu.gr/~asidirop/delay.php > download1
[2]-  Running                 curl -s -N https://users.iee.ihu.gr/~asidirop/delay.php > download2 &
```

   Δώστε την εντολή

```
bg %1
```

Δηλαδή % και τον αριθμό του job που αντιστοιχεί στο σταματημένο job.

Η εντολή bg στέλνει το σήμα CONT στην διεργασία που αντιστοιχεί, και την βάζει να εκτελείται στο παρασκήνιο.

Εκτελέστε πάλι την εντολή jobs.

Δώστε την εντολή

```
fg %1
```

Η εντολή fg στέλνει το σήμα CONT στην διεργασία που αντιστοιχεί, και την βάζει να εκτελείται στο προσκήνιο, δηλαδή της δίνει τον έλεγχο του τερματικού.

Πατήστε τα πλήκτρα Cntrl-C (συμβολίζεται ^C).

Το ^C στέλνει στην τρέχουσα διεργασία το σήμα TERM.

Η εφαρμογή τερματίστηκε. Τώρα ως job έχουμε μόνο το curl στο download2.

Ανοίξτε ακόμη ένα τερματικό στο users. Δείτε τις διεργασίες που εκτελούνται. Μια από αυτές είναι το curl.

Κλείστε το τερματικό από το οποίο τρέξατε τα προηγούμενα jobs (πατώντας στο παράθυρο το κουμπί για κλείσιμο).

Το bash, όταν τερματίζεται, τερματίζει και όλα τα jobs του (στέλνει σε όλα τα jobs το σήμα SIGHUP).

Εάν δεν θέλουμε ένα job να μην τερματιστεί με την έξοδο από το shell, τότε πρέπει να δώσουμε την εντολή disown %x (όπου x το job number). Έτσι θα συνεχίσει να εκτελείται το curl ακόμη και μετά το κλείσιμο του putty.

Η εντολή disown αποδεσμεύει ένα job από το τρέχον shell. Η διεργασία όμως παραμένει να τρέχει.

ΑΣΚΗΣΗ 3

Κάθε διεργασία έχει 3 προκαθορισμένα κανάλια εισόδου-εξόδου (I/O streams).

Το input, output και error (στην C συμβολίζονται ως stdin, stdout, stderr. Στην C++ συμβολίζονται ως cin, cout, cerr. Στην java συμβολίζονται ως System.in, System.out, System.err. Σε όλες τις γλώσσες προγραμματισμού έχουν κάποιον συμβολισμό.) Εσωτερικά στο πρόγραμμα όμως, όλα αντιστοιχούν σε αριθμούς. Το stdin αντιστοιχεί πάντα στο 0, το stdout πάντα στο 1, το stderr πάντα στο 2.

Όταν ένα πρόγραμμα (διεργασία) θέλει να διαβάσει από το πληκτρολόγιο, τότε διαβάζει από το stdin.

Όταν ένα πρόγραμμα (διεργασία) θέλει να τυπώσει κάτι στην οθόνη, τότε τυπώνει στο stdout.

Όταν ένα πρόγραμμα (διεργασία) θέλει να τυπώσει ένα μήνυμα λάθους, τότε τυπώνει στο stderr.

Αυτό ισχύει για όλες τις γλώσσες προγραμματισμού και για όλα τα λειτουργικά συστήματα (σχεδόν).

Για κάθε διεργασία (κατά την δημιουργία της) μπορούμε να ορίσουμε εμείς σε τι θα αντιστοιχούν αυτά τα I/O streams (αν δεν το ορίσουμε, τότε είναι το τερματικό). Για το ορισμό αυτό χρησιμοποιούμε τα σύμβολα: >, <, >>, <<, …

Η σύνταξη είναι:

Εντολή > file

Εντολή > file1 < file2 2> file3

>file, 1> file	Άλλαξε το stdout να «γράφει» στο αρχείο file και όχι στο προκαθορισμένο. Το file αν υπάρχει θα διαγραφεί, αν δεν υπάρχει θα δημιουργηθεί.
>>file, 1>> file	Άλλαξε το stdout να «γράφει» στο αρχείο file και όχι στο προκαθορισμένο. Το file αν υπάρχει ΔΕΝ θα διαγραφεί, αν δεν υπάρχει θα δημιουργηθεί. Τα δεδομένα από την εντολή θα γίνουν append στο file.
2> file	Άλλαξε το stderr να «γράφει» στο αρχείο file και όχι στο προκαθορισμένο. Το file αν υπάρχει θα διαγραφεί, αν δεν υπάρχει θα δημιουργηθεί.
2>> file	Άλλαξε το stderr να «γράφει» στο αρχείο file και όχι στο προκαθορισμένο. Το file αν υπάρχει ΔΕΝ θα διαγραφεί, αν δεν υπάρχει θα δημιουργηθεί. Τα δεδομένα από την εντολή θα γίνουν append στο file.
< file	Άλλαξε το stdin να «διαβάζει» από το αρχείο file και όχι από το προκαθορισμένο (τερματικό). Αν δεν υπάρχει το file τότε θα πάρουμε σφάλμα.
<< STRING	Άλλαξε το stdin να «διαβάζει» από το shell. Το shell θα περιμένει να πληκτρολογήσουμε δεδομένα μέχρι να πληκτρολογήσουμε την γραμμή STRING. Τότε, ό,τι πληκτρολογήσαμε θα το στείλει στο stdin της διεργασίας.

Να βρεθεί το αποτέλεσμα της εκτέλεσης των παρακάτω εντολών (συνδεθείτε στο users.it.teithe.gr):

1. `who > file1`

```
2.    cat file1
3.    wc file1
4.    wc < file1
5.    ls -l file1
6.    echo "Hello world" > file2
7.    more file2
8.    wc file2
9.    sort < file1 > file3
10.   cat file3
```

Ποια είναι η διαφορά της εντολής 3 από την 4 ?

Οι εντολές wc, grep, sort (και όλες οι αντίστοιχες) συνήθως ονομάζονται εντολές φίλτρα. Αυτές οι εντολές (εάν δεν τις δώσουμε ως όρισμα όνομα αρχείου) διαβάζουν δεδομένα από το stdin και τυπώνουν τα αποτελέσματα στο stdout.

Δώστε την εντολή:

wc

θα παρατηρήσετε ότι το τερματικό «κόλλησε»…. Όχι δεν κόλλησε. Η διεργασία που αντιστοιχεί στην εντολή περιμένει να διαβάσει από το stdin. Το stdin (εφόσον δεν ορίσαμε διαφορετικά είναι το τερματικό). Μόνο που η εντολή δεν μας εμφανίζει κάποιο prompt (όπως το shell) για να το αντιληφθούμε. Πληκτρολογήστε κείμενο. Πότε θα σταματήσει να διαβάζει η διεργασία?

Όταν ένα πρόγραμμα διαβάζει από το stdin, διαβάζει με τον ίδιο ακριβώς τρόπο με τον οποίο θα διάβαζε δεδομένα από ένα αρχείο. Αν μέσα από ένα πρόγραμμα θα θέλαμε να διαβάσουμε όλα τα περιεχόμενα ενός αρχείου, θα διαβάζαμε (byte-byte ή γραμμή-γραμμή) μέχρι να φτάσουμε στο τέλος του αρχείου. Όταν διαβάζοντας ένα αρχείο φτάσουμε στο τέλος του και δεν υπάρχουν άλλα δεδομένα να διαβάσουμε, τότε το λειτουργικό στέλνει έναν ειδικό χαρακτήρα στο πρόγραμμά μας που ονομάζεται “End-Of-File” και συνήθως συμβολίζεται με το EOF. Τότε το πρόγραμμα αντιλαμβάνεται ότι δεν υπάρχει κάτι άλλο να διαβάσει από το αρχείο.

Το ίδιο κάνει και η εντολή wc όταν διαβάζει από τον stdin (ή και όταν διαβάζει από αρχείο - είναι η ίδια συμπεριφορά). Διαβάζει συνεχώς μέχρι να φτάσει στον ειδικό χαρακτήρα EOF. Αυτόν τον ειδικό χαρακτήρα, μπορούμε να τον δημιουργήσουμε από το πληκτρολόγιο (και άρα να τον διαβάσει η εντολή) πατώντας Cntrl-D (^D).

Μόλις πατήσουμε ^D η διεργασία θα σταματήσει να διαβάζει και θα μας εμφανίσει το αποτέλεσμα.

Δώστε την εντολή:

```
grep test
```

Η εντολή grep θα αναζητήσει το string “test” πού? Εφόσον δεν ορίσαμε αρχείο, τότε θα διαβάσει από την κανονική είσοδο (stdin), δηλαδή το τερματικό. Πληκτρολογήστε μερικές γραμμές (ανάμεσα σε αυτές και κάτι που να περιέχει το test). Τι παρατηρείτε? Πατήστε ^D ώστε να ολοκληρωθεί η εντολή.

ΑΣΚΗΣΗ 4

Να βρεθεί το αποτέλεσμα της εκτέλεσης των παρακάτω εντολών :

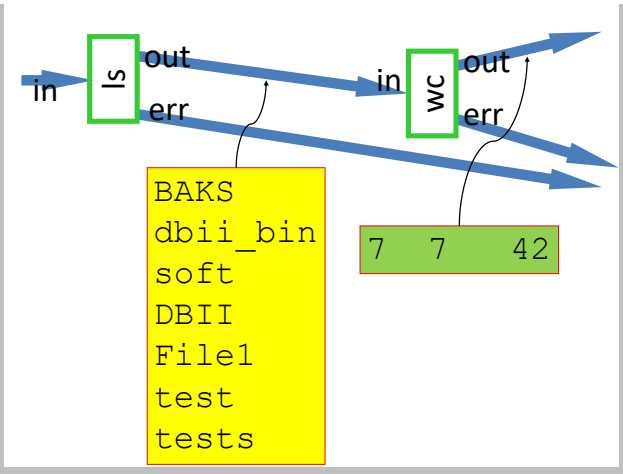
1. `ls | wc -l`
2. `who | wc -l`
3. `ls *.c | wc -l`
4. `who | sort | more`

Με τον χαρακτήρα | (διασωλήνωση - pipe) μπορούμε να δημιουργήσουμε μια «σωλήνα» (pipe) μεταξύ 2 διεργασιών. Για κάθε | που χρησιμοποιούμε, δημιουργείται (κάπου στον δίσκο) ένα αρχείο τύπου FIFO (first in – first out) ή αν θέλετε τύπου pipe. Σε αυτά τα αρχεία μόνο ένας (μια διεργασία) μπορεί να γράψει, και μόνο μια διεργασία μπορεί να διαβάσει. Οτιδήποτε διαβάζεται, χάνεται (σβήνεται) από το αρχείο. Τα δεδομένα υποχρεωτικά διαβάζονται (από την διεργασία που διαβάζει) με την ίδια ακριβώς σειρά με την οποία γράφτηκαν (First in- first out) από την διεργασία που γράφει.

Με την εντολή:

ls | wc

δημιουργούμε 2 διεργασίες. Η πρώτη θα έχει ως stdout ένα αρχείο pipe. Η $2^{η}$ θα έχει ως stdin το ίδιο αρχείο. Έτσι ότι γράψει η $1^{η}$ διεργασία στο stdout θα πάει στο stdin της $2^{ης}$. Σχηματικά μπορούμε να το δούμε ως:



Προσοχή, με την διασωλήνωση αλλάζει μόνο το stdout της εντολής που βρίσκεται στα αριστερά και όχι το stderr. Το stderr θα παραμείνει ως έχει, δηλαδή το τερματικό.

Εκτελέστε τις εντολές:

```
5. ls /tmp | wc -l
6. ls /ttt | wc -l
```

Τι μετράει h wc στην 2η περίπτωση? Που εμφανίζεται το σφάλμα?

ΑΣΚΗΣΗ 5

Χρησιμοποιώντας διασωλήνωση:

1. Βρείτε τα αρχεία του τρέχοντος καταλόγου (μόνο τα αρχεία, όχι τους υποκαταλόγους) και εμφανίστε τα ταξινομημένα με βάση το μέγεθος σε φθίνουσα σειρά (δείτε το man της εντολής sort).
2. Εμφανίστε τα περιεχόμενα του αρχείου /etc/passwd με κεφαλαίους χαρακτήρες. (δείτε το man της εντολής tr)
3. Βρείτε τα αρχεία (και καταλόγους) που το όνομά τους ξεκινάει με d, και εμφανίστε τα ονόματά τους με κεφαλαία γράμματα σε φθίνουσα αλφαβητική σειρά.
4. Μετρήστε πόσοι κατάλογοι υπάρχουν στον τρέχον φάκελο.
5. Μετρήστε πόσοι χρήστες υπάρχουν στο σύστημα (στο users μετρήστε τις γραμμές από την εντολή getent passwd).
6. Μετρήστε πόσοι χρήστες έχουν username που ξεκινάει με a και τελειώνει σε p. (13)
7. Βρείτε την διεργασία με το μεγαλύτερο PID. (θυμηθείτε τις εντολές sort, head, tail)
8. Βρείτε το αρχείο (από τον τρέχον φάκελο) με το μεγαλύτερο μέγεθος.

Σενάρια φλοιού

Για την δημιουργία ενός shell script, αρκεί να φτιάξουμε ένα αρχείο κειμένου (ASCII) το οποίο να περιέχει τις εντολές μας.

Συνήθως τα αρχεία που είναι σενάρια φλοιού, είτε το όνομά τους δεν έχει καμιά κατάληξη (πχ: myscript) είτε βάζουμε την κατάληξη .sh (πχ: myscript.sh).

Μέσα στο αρχείο πρέπει να εισάγουμε ως πρώτη γραμμή το:

#!/bin/bash

(ή #!/bin/sh ανάλογα με το αν χρησιμοποιούμε τις επεκτάσεις του bash έναντι του sh. Είναι προτιμότερο το bash καθώς το sh τείνει να καταργηθεί η χρήση του).

Αυτή η πρώτη γραμμή δίνει την πληροφορία στον πυρήνα ποιος είναι ο interpreter που θα εκτελέσει τις εντολές που περιέχονται στο αρχείο.

Παράδειγμα ενός απλού script:

```
#!/bin/sh


echo "This is my first script"

echo -n "I am the user: "

whoami
```

Εάν τα παραπάνω τα αποθηκεύσουμε στο αρχείο myscript, και αφού δώσουμε το δικαίωμα execute στο αρχείο (με την εντολή chmod), μπορούμε να το εκτελέσουμε αν πληκτρολογήσουμε:

```
./myscript
```

ΑΣΚΗΣΗ 6

Φτιάξτε ένα σενάριο κελύφους με όνομα myls, το οποίο θα εκτελεί την εντολή ls -l (θα εμφανίζει τα αποτελέσματά της), και μετά θα εμφανίζει το πλήθος των απλών αρχείων, το πλήθος των καταλόγων, το πλήθος των κρυφών αρχείων, το πλήθος των κρυφών καταλόγων.

ΑΣΚΗΣΗ 7

Δημιουργήστε ένα script με όνομα echo_test το οποίο να περιέχει:

```
#!/bin/sh
#echo_test
#----------------------------------
echo "1.the process id is  :    $$ and cwd $PWD"
echo '2.the process id is  :    $$ and cwd $PWD '
echo  3.the process id is  :    $$ and cwd $PWD
echo  4.the process id is\ \ :\ \ \ \$\$ and cwd $PWD
echo '5.the process id is  :    "$$ and cwd $PWD" '
echo "6.the process id is  :    '$$' and cwd $PWD"
```

Τι παρατηρείτε;

ΑΣΚΗΣΗ 8

Δημιουργήστε ένα script με όνομα echo_test3 το οποίο να περιέχει:

```
#!/bin/bash
#echo_test3
#----------------------------------
a=TEST
b=TEST   B
c="TEST   C"
d="$c   +   D"
echo "1.a is $a"
echo
echo "2.b is $b"
echo
echo '3.c is $c'
echo "4.c is $c"
echo 5.c is $c
echo
echo "5.d is $d"
echo
var1="6 "
var2=$((var1 + 2 ))
echo "7.var1 is $var1, var2 is $var2"
```

1. Τι παρατηρείτε; Ποιες είναι οι διαφορές με την χρήση " ή '
2. Η μεταβλητή B, τι τιμή έχει?

ΑΣΚΗΣΗ 9

Δημιουργήστε 3 scripts με όνομα echo_test4(a,b,c) τα οποία να περιέχουν:

Echo_test4a

```
#!/bin/sh
#echo_test4a
#----------------------------------
x=`ls -l`
```

```
echo x is $x
```

Echo_test4b

```
#!/bin/sh
#echo_test4b
#-----------------------------------
x=`ls -l`
echo  'x is $x'
```

Echo_test4c

```
#!/bin/sh
#echo_test4c
#-----------------------------------
x=`ls -l`
echo  "x is $x"
```

Τα ανάποδα εισαγωγικά (`) προκαλούν εκτέλεση της εντολής που περικλείουν. Οι εντολές που περικλείονται στα (`) θα εκτελεστούν, όμως η κανονική τους έξοδος δεν θα εμφανιστεί στην οθόνη, θα δεσμευτεί και τα επιστραφεί στην μεταβλητή (εφόσον τα (`) χρησιμοποιήθηκαν σε εκχώρηση.

ΑΣΚΗΣΗ 10

Δημιουργήστε ένα script με όνομα echo_test5 το οποίο να περιέχει:

```
#!/bin/bash
#echo_test5
#-----------------------------------
a=5
b=" 6"
c=""
d=$((a+5))
e=$((b+5))
f=$((c+5))

echo "d is $d, e is $e, f is $f"
echo

d=`expr $a + 5`
e=`expr $b+5`
f=`expr $c+5`

echo "d is $d, e is $e, f is $f"
```

ΑΣΚΗΣΗ 11

Φτιάξτε το παρακάτω script:

```
#!/bin/sh
#script06
#---------------------------------
a=5
echo -n "Give me your name: "
read name
echo "value of a is $a"
echo "value of name is $name"
```

ΑΣΚΗΣΗ 12

Τι θα εμφανιστεί στην οθόνη με την εκτέλεση του παρακάτω προγράμματος κελύφους :

```
#!/bin/sh
#script07
#---------------------------------
echo "first parameter : $1"
echo "third parameter : $3"
echo "ninth parameter : $9"
echo "tenth parameter : $10"
echo "eleventh parameter : $11"
echo "No of parameters : $#"
echo "all parameters : $*"
echo "all parameters : $@"
shift
echo "first parameter : $1"
echo "all parameters : $*"
```

με την γραμμή εντολής

```
./script07 one two 3 4 5 6 7 eight 9 ten 11
```

Προ-δηλωμένες μεταβλητές:

$1,$2,…$9 : οι τιμές των 9 πρώτων ορισμάτων που έδωσε ο χρήστης σε ένα script

$# : το πλήθος των ορισμάτων που έδωσε ο χρήστης σε ένα script

$*, $@: Οι πίνακες με τα ορίσματα που έδωσε ο χρήστης σε ένα script

$$: Το ProcessID (PID) του τρέχοντος shell