ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2016-2017

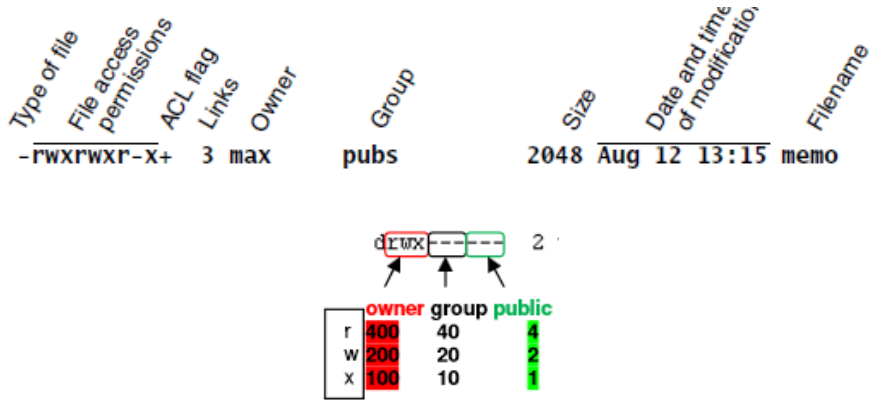
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ

ΜΑΘΗΜΑ: ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΤΗΡΙΟ 2 - ΣΗΜΕΙΩΣΕΙΣ

Άδειες Χρήσης Αρχείων και Καταλόγων

Το Linux υποστηρίζει έλεγχο πρόσβασης σε αρχεία. Μπορούμε να δούμε τις άδειες χρήσης κάθε αρχείου και καταλόγου εφαρμόζοντας της εντολή ls –l. Η ακόλουθη εικόνα επεξηγεί τη σημασία του κάθε χαρακτήρα ελέγχου.



Οι άδειες πρόσβασης είναι οι ακόλουθες:

- r : ανάγνωση (τιμή 4)
- w : εγγραφή (τιμή 2)
- x : εκτέλεση (τιμή 1)

Η αλλαγή των αδειών γίνεται με την εντολή chmod. Παραδείγματα:

- chmod a+rw letter : προσθέτει τις άδειες ανάγνωσης και εγγραφής σε όλους τους χρήστες για το αρχείο letter.
- chmod o-rw letter : αφαιρεί τις άδειες ανάγνωσης και εγγραφής στους άλλους χρήστες για το αρχείο letter.

Οι αναφορές σε χρήστες γίνονται ως ακολούθως:

- u : ο χρήστης / ιδιοκτήτης του αρχείου
- o : οι άλλοι χρήστες
- a : όλοι οι χρήστες
- g : η ομάδα του ιδιοκτήτη

Στην chmod μπορούμε να εισάγουμε και αριθμητικές αναφορές στις άδειες χρήσης. Παραδείγματα:

- chmod 600 letter : καθορίζει τις άδειες ανάγνωσης και εγγραφής στον ιδιοκτήτη του αρχείου.
- chmod 604 letter : καθορίζει τις άδειες ανάγνωσης και εγγραφής στον ιδιοκτήτη του αρχείου και τη άδεια ανάγνωσης στους υπόλοιπους χρήστες.
- chmod 777 letter : καθορίζει όλες τις άδειες για όλους τους χρήστες.

Με την εντολή **chown** μπορούμε να αλλάξουμε τον ιδιοκτήτη ενός αρχείου. Για την εκτέλεση της εντολής χρειαζόμαστε να έχουμε δικαιώματα superuser. Παραδείγματα:

- chown bob test : Τοποθετεί τον bob ως ιδιοκτήτη τους αρχείου test.
- chown bob:users test : Τοποθετεί τον bob ως ιδιοκτήτη τους αρχείου test και την ομάδα του ίδιου αρχείου σε users.
- chown :users test : Τοποθετεί την ομάδα του αρχείου test σε users αφήνοντας τον ιδιοκτήτη τον ίδιο.

Εντολή umask

Η εντολή umask μας επιτρέπει να ελέγξουμε τις προκαθορισμένες ρυθμίσεις που αφορούν στις άδειες των αρχείων/καταλόγων. Χρησιμοποιεί οκταδικές αναπαραστάσεις για να καθορίσει ένα σύνολο bits που απεικονίζουν τη μάσκα που θα χρησιμοποιηθεί για τον καθορισμό των αδειών. Παραδείγματα:

- umask : μας δείχνει την τρέχουσα τιμή της μάσκας (π.χ., 0002, 0022).
- umask 0022 : καθορίζει τις άδειες των αρχείων σε 644 (666 - 022) και των καταλόγων σε 755 (777 – 022).
- umask –S : εμφανίζει τη μάσκα για τις άδειες των αρχείων με συμβολικούς όρους.
- umask u=rw,go= : για κάθε νέο αρχείο ο ιδιοκτήτης θα έχει άδειες ανάγνωσης και εγγραφής ενώ η ομάδα του και οι υπόλοιποι χρήστες δεν θα έχουν καμία άδεια πάνω στο αρχείο.

Προσοχή: η μάσκα συμβολίζει τις άδειες που δεν επιτρέπονται. Ο ακόλουθος πίνακας παρουσιάζει τις άδειες που προκύπτουν από τα διάφορα ψηφία που υιοθετούνται στην εντολή umask.

umask digit	resulting default file permissions	resulting default directory permissions
0	rw	rwx
1	rw	rwx
2	r	rx
3	r	r
4	w	wx
5	w	w
6	x	x
7	(no permission allowed)	(no permission allowed)

ΣΗΜΕΙΩΣΗ: Για ένα αρχείο μπορούμε να αλλάξουμε / τροποποιήσουμε τις άδειες και γενικά τις ιδιότητές του στο γραφικό περιβάλλον του Linux κάνοντας δεξί κλικ πάνω στο αρχείο και επιλέγοντας Properties. Στη συνέχεια μπορούμε να αλλάξουμε τις άδειες χρήσης του αρχείου μέσω της καρτέλας Permissions καθώς και το πρόγραμμα με το οποίο ανοίγουμε το αρχείο στην καρτέλα Open with.

Τέλος, όλες οι εντολές διαχείρισης αρχείων τις οποίες έχουμε ήδη δει και εκτελέσει μέσω του terminal (π.χ., cp, mv, mkdir, etc.) μπορούμε να τις εκτελέσουμε στο παραθυρικό περιβάλλον του Linux με χρήση των λειτουργιών του ποντικιού αλλά και μέσω συντομεύσεων του πληκτρολογίου.

Εργασία με Χρήστες

Το Linux προσφέρει ένα σύνολο εντολών για τη διαχείριση των χρηστών. Κάποιες από αυτές είναι οι ακόλουθες:

Εντολή passwd

Η εντολή passwd επιτρέπει την αλλαγή του συνθηματικού εισόδου για κάποιο χρήστη. Η ακόλουθη εικόνα μας δείχνει ένα παράδειγμα αλληλεπίδρασης κατά την εκτέλεση της εντολής.

```
kk@ubuntu:~$ passwd
Changing password for kk.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Αν έχετε δικαιώματα superuser και σαν όρισμα της εντολής θέσετε το username ενός άλλου χρήστη, τότε το Linux θα σας δώσει τη δυνατότητα να αλλάξετε το συνθηματικό του συγκεκριμένου χρήστη.

Εντολή who

Η εντολή who εμφανίζει τους χρήστες που είναι συνδεδεμένοι στο σύστημα. Η ακόλουθη οθόνη παρουσιάζει ένα παράδειγμα εκτέλεσης της εντολής.

```
$ who
sam      tty4         2009-07-25 17:18
max      tty2         2009-07-25 16:42
zach     tty1         2009-07-25 16:39
max      pts/4        2009-07-25 17:27 (coffee)
```

Εντολή finger

Η εντολή finger εμφανίζει τους χρήστες που είναι συνδεδεμένοι στο σύστημα αλλά μπορεί και να χρησιμοποιηθεί ώστε να πάρουμε πληροφορίες για κάποιο χρήστη. Η ακόλουθη οθόνη παρουσιάζει ένα παράδειγμα εκτέλεσης της εντολής.

```
$ finger max
Login: max                              Name: Max Wild
Directory: /home/max                    Shell: /bin/tcsh
On since Fri Jul 25 16:42 (PDT) on tty2 (messages off)
On since Fri Jul 25 17:27 (PDT) on pts/4 from coffee
   3 minutes 7 seconds idle
New mail received Sat Jul 25 17:16 2009 (PDT)
     Unread since Sat Jul 25 16:44 2009 (PDT)
Plan:
I will be at a conference in Hawaii all next week.
If you need to see me, contact Zach Brill, x1693.
```

Ο ακόλουθος πίνακας παρουσιάζει μια σύγκριση ανάμεσα στις εντολές w, who και finger.

Information displayed	w	who	finger
Username	x	x	x
Terminal-line identification (tty)	x	x	x
Login time (and day for old logins)	x		
Login date and time		x	x
Idle time	x		x
Program the user is executing	x		
Location the user logged in from			x
CPU time used	x		
Full name (or other information from **/etc/passwd**)			x
User-supplied vanity information			x
System uptime and load average	x		

Εντολή write

Η εντολή write στέλνει ένα μήνυμα στο χρήστη που επιθυμούμε. Με Ctrl-D τερματίζουμε την αποστολή του μηνύματος.

Εντολή msg

Η εντολή msg ενεργοποιεί/απενεργοποιεί τη λήψη μηνυμάτων από τους άλλους χρήστες. Παραδείγματα:

- msg y : ενεργοποίηση της λήψης μηνυμάτων.
- msg n : απενεργοποίηση της λήψης μηνυμάτων.

Εντολή su

Η εντολή su μας επιτρέπει να ξεκινήσουμε το φλοιό σαν κάποιος άλλος χρήστης. Εφόσον χρησιμοποιηθεί η επιλογή –l τότε το session ξεκινά από μια login page ώστε να ξεκινήσει η σύνδεση από την αρχή. Κατά τη σύνδεση αλλάζει και το working directory το οποίο γίνεται το home directory του νέου χρήστη. Αν δεν ορίσουμε το όνομα του νέου χρήστη, το Linux υποθέτει πως ο νέος χρήστης είναι ο superuser.

Εντολή sudo

Η εντολή sudo μας επιτρέπει να εκτελέσουμε μια εντολή ως ένας άλλος χρήστης. Συνήθως, η εντολή χρησιμοποιείται ώστε να εκτελέσουμε εντολές ως superuser. Παραδείγματα:

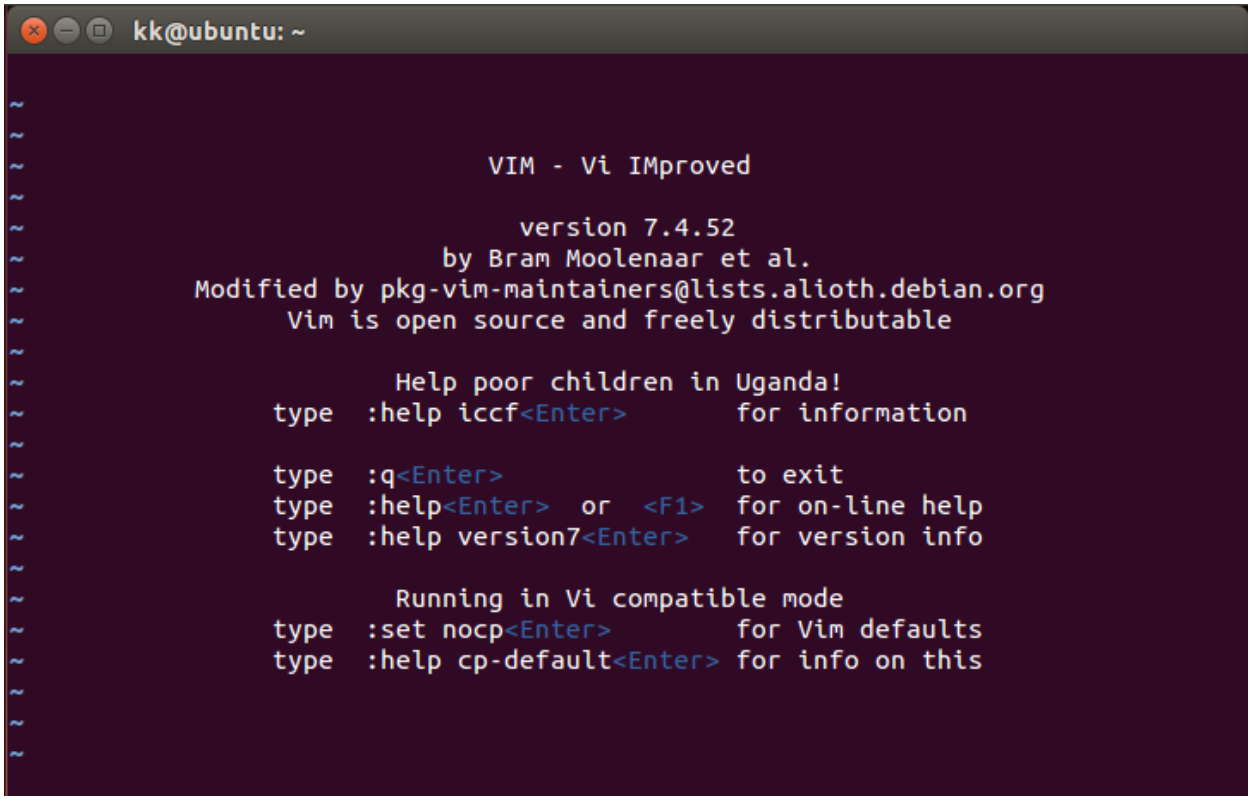
- sudo apt-get install finger : ξεκινά το κατέβασμα και εγκατάσταση του εργαλείου finger.
- sudo backup_script : Ξεκινά την εκτέλεση ενός backup script.

Κατά την εκτέλεση της εντολής sudo θα ζητηθεί το συνθηματικό του νέου χρήστη.

Εργασία με συντάκτες κειμένου

Α. Ο Συντάκτης κειμένου vi

Ο vi είναι ένας συντάκτης κειμένου πολύ γρήγορος πάντα διαθέσιμος σε συστήματα Linux/Unix. Για να ξεκινήσουμε το συντάκτη απλά πληκτρολογούμε vi [όνομα_αρχείου]. Η οθόνη που εμφανίζεται είναι η ακόλουθη (vmware - Ubuntu):



Για να βγούμε από το συντάκτη vi πιέζουμε Shift + Q και μεταβαίνουμε στο prompt : (Ex mode). Εκεί μπορούμε να αποθηκεύσουμε το αρχείο με την εντολή w [όνομα_αρχείου] και να βγούμε στο prompt του terminal με την εντολή q. Η εντολή q! θα προκαλέσει 'βίαιη' έξοδο από τον vi χωρίς την αποθήκευση του αρχείου ή των αλλαγών που έχουν γίνει. Για να επιστρέψουμε στη διαχείριση του κειμένου, μπορούμε στο Ex mode (:) να δώσουμε την εντολή visual.

Στον vi δουλεύουμε σε δύο modes: εισαγωγή εντολών που σχετίζονται με την διαχείριση και τη μετακίνηση στα περιεχόμενα του αρχείου ή σε εισαγωγή κειμένου όπου εισάγουμε το κείμενο που επιθυμούμε. Το πέρασμα από την εισαγωγή κειμένου σε εισαγωγή εντολών γίνεται με τη χρήση του πλήκτρου Esc ενώ το πέρασμα σε εισαγωγή κειμένου γίνεται με τη χρήση των εντολών που αναφέρονται πιο κάτω.

Συνοπτικά, οι εντολές που μπορούν να χρησιμοποιηθούν στο περιβάλλον του vi έχουν ως ακολούθως:

Cursor Movement

- h move left (backspace)
- j move down
- k move up
- l move right (spacebar)
- [return] move to the beginning of the next line
- $ last column on the current line
- 0 move cursor to the first column on the current line
- ^ move cursor to first nonblank column on the current line
- w move to the beginning of the next word or punctuation mark
- W move past the next space
- b move to the beginning of the previous word or punctuation mark
- B move to the beginning of the previous word, ignores punctuation
- e end of next word or punctuation mark
- E end of next word, ignoring punctuation
- H move cursor to the top of the screen

M move cursor to the middle of the screen
L move cursor to the bottom of the screen

Screen Movement

G move to the last line in the file
xG move to line x
z+ move current line to top of screen
z move current line to the middle of screen
z- move current line to the bottom of screen
^F move forward one screen
^B move backward one line
^D move forward one half screen
^U move backward one half screen
^R redraw screen (does not work with VT100 type terminals)
^L redraw screen (does not work with Televideo terminals)

Inserting

r replace character under cursor with next character typed
R keep replacing character until [esc] is hit
i insert before cursor
a append after cursor
A append at end of line
O open line above cursor and enter append mode

Deleting

x delete character under cursor
dd delete line under cursor
dw delete word under cursor
db delete word before cursor

Find Commands

? finds a word going backwards
/ finds a word going forwards
f finds a character on the line under the cursor going forward
F finds a character on the line under the cursor going backwards
t find a character on the current line going forward and stop one character before it
T find a character on the line going backward and stop one character before it
; repeat last f, F, t, T

Miscellaneous Commands

. repeat last command
u undoes last command issued
U undoes all commands on one line
xp deletes first character and inserts after second (swap)
J join current line with the next line
^G display current line number
% if at one parenthesis, will jump to its mate
mx mark current line with character x
'x find line marked with character x

Line Editor Mode

Any commands form the line editor ex can be issued upon entering line mode.
To enter: type ':'
To exit: press[return] or [esc]

:w saves the current file without quitting
:# move to line #
:$ move to last line of file

Β. Συντάκτες με γραφικό περιβάλλον

Το Linux, πέρα από τους συντάκτες κειμένου που έχει 'κληρονομήσει' από το Unix, προσφέρει και συντάκτες κειμένου που προσφέρουν γραφικό περιβάλλον εργασίας. Κάποιοι από αυτούς είναι: gedit, Kedit, NEdit, Tea, etc.

Για τη χρήση των συντακτών αυτών το μόνο που έχουμε να κάνουμε (μια από τις επιλογές μας) είναι να επιλέξουμε, στο γραφικό περιβάλλον του Linux, να μπούμε στο home directory μας και να δημιουργήσουμε ένα κενό αρχείο ως εξής: Δεξί κλικ → New Document → Empty Document. Στη συνέχεια δίνουμε το όνομα του αρχείου και πιέζουμε Enter. Έπειτα με διπλό κλικ πάνω στο νέο αρχείο θα ανοίξει ο συντάκτης κειμένου gedit. Τέλος, μπορούμε να εργαστούμε στο γραφικό περιβάλλον χρησιμοποιώντας όλες τις δυνατότητες ενός τυπικού συντάκτη κειμένου.

Εξάσκηση

Στο ακόλουθο link θα βρείτε ένα πλήθος συντακτών που υποστηρίζονται στο Linux. http://www.yolinux.com/TUTORIALS/LinuxTextEditors.html

Μπορείτε να εκτελέστε κάποιους από αυτούς και να τους δοκιμάσετε. Αν κάποιος δεν είναι εγκατεστημένος μπορείτε να τον εγκαταστήσετε με την εντολή sudo apt-get install <package_name>.

Στο βιβλίο του Sobell, στα κεφάλια 6, 7 περιγράφεται η χρήση των συντακτών emacs και vim.

Εισαγωγή στον Προγραμματισμό Φλοιού

Α. Εισαγωγή

Στην πιο απλή εκδοχή ένα πρόγραμμα φλοιού είναι ένα αρχείο το οποίο περιέχει ένα σύνολο εντολών. Ο φλοιός διαβάζει το αρχείο και εκτελεί τις εντολές σαν να έχουν αποδοθεί από τη γραμμή εντολών. Η διαδικασία συγγραφής ενός προγράμματος φλοιού στο Linux είναι πολύ απλή. Με ένα συντάκτη κειμένου δημιουργούμε ένα αρχείο στο οποίο τοποθετούμε τις εντολές που επιθυμούμε να εκτελεστούν. Στη συνέχεια κάνουμε το αρχείο εκτελέσιμο και τοποθετούμε το script σε σημείο που να μπορεί να το 'βρει' ο φλοιός.

Στο home directory μας δημιουργούμε το αρχείο bash_script με τα ακόλουθα περιεχόμενα:

```
#! /bin/bash
# This is our first script
echo "Hello World‼"
```

Η πρώτη γραμμή του script υποδεικνύει στο φλοιό πως το script θα εκτελεστεί με τη βοήθεια του φλοιού bash (***Προσοχή***: το #! έχει ειδική σημασία για το φλοιό και ονομάζεται shebang) ενώ η δεύτερη γραμμή περιλαμβάνει σχόλια τα οποία φυσικά αγνοούνται από το φλοιό. Η τρίτη γραμμή περιλαμβάνει μια απλή εντολή echo.

Στη συνέχεια εκτελούμε την εντολή chmod +x bash_script ώστε το αρχείο μας να γίνει εκτελέσιμο. Τέλος, με την εντολή ./bash_scipt μπορούμε να εκτελέσουμε το script. Εναλλακτικά, μπορούμε να δώσουμε την ακόλουθη εντολή για την εκτέλεση του script: bash bash_script.

Προσοχή. Πριν το όνομα του script πρέπει να παραθέσουμε τη διαδρομή ώστε να μπορέσει ο φλοιός να βρει και να εκτελέσει το script. Σε διαφορετική περίπτωση ο φλοιός θα αναζητήσει το script στους προκαθορισμένους καταλόγους π.χ., στον κατάλογο /bin.

Το δεύτερο script θα μας παρουσιάσει μια λίστα με τις ενεργές διεργασίες. Με ένα συντάκτη κειμένου δημιουργήστε το αρχείο show_processes με τα ακόλουθα περιεχόμενα:

```
#! /bin/bash
echo "Hello $USER"
echo "Hey i am" $USER "and will be telling you about the current processes"
echo "Running processes List"
ps
```

Το script μετά την προσφώνηση με το username μας θα εμφανίσει τη λίστα με τις ενεργές διεργασίες του συστήματος (θα μιλήσουμε περισσότερο για την εντολή ps σε επόμενο εργαστήριο).

Β. Μεταβλητές

Οι μεταβλητές μας δίνουν τη δυνατότητα να εμπλέξουμε αλληλεπίδραση του χρήστη με το φλοιό. Για παράδειγμα, σε command line δώστε τις ακόλουθες εντολές:

$ foo=”Test”

$ echo $foo

Σε αυτό το παράδειγμα, δημιουργούμε μια νέα μεταβλητή με όνομα foo στην οποία τοποθετούμε την τιμή Test. Η εντολή echo μας παρουσιάζει την τιμή της μεταβλητής foo στην οθόνη (Προσοχή. Χρησιμοποιούμε το $ για να προσπελάσουμε την τιμή μιας μεταβλητής). Η σύνταξη για την εκχώρηση τιμής σε μια μεταβλητή έχει ως εξής:

Variable=value

Σημείωση: Οι μεταβλητές περιβάλλοντος προσπελαύνονται με το όνομά τους γραμμένο με κεφαλαία γράμματα π.χ., HOME, PATH.

Εντολή unset

Μπορούμε να διαγράψουμε μεταβλητές με χρήση της εντολής unset.

Παράδειγμα: unset foo.

Εντολή readonly

Με την εντολή readonly ορίζουμε ότι μια μεταβλητή δεν θα αλλάξει τιμή.

Παράδειγμα: readonly foo

Εντολή declare

Η εντολή declare χρησιμοποιείται για να καθορίσει τον τύπο μιας μεταβλητής. Ο ακόλουθος πίνακας συνοψίζει ένα σύνολο παραμέτρων της εντολής καθώς και τη σημασία τους.

Attribute	Meaning
–a	Declares a variable as an array
–f	Declares a variable to be a function name
–i	Declares a variable to be of type integer
–r	Makes a variable readonly; also readonly
–x	Exports a variable (makes it global); also export

Παραδείγματα:

$ declare –r foo

$ declare –i foo

$ declare person=kk

Ειδικοί Χαρακτήρες

Ο ακόλουθος πίνακας παρουσιάζει ένα σύνολο χαρακτήρων που έχουν ειδική σημασία για το φλοιό και μπορούν να χρησιμοποιηθούν στα scripts.

Character	Use
NEWLINE	Initiates execution of a command
;	Separates commands
( )	Groups commands for execution by a subshell or identifies a function
(( ))	Expands an arithmetic expression
&	Executes a command in the background
|	Sends standard output of the preceding command to standard input of the following command
>	Redirects standard output
>>	Appends standard output
<	Redirects standard input
<<	Here document
*	Any string of zero or more characters in an ambiguous file reference
?	Any single character in an ambiguous file reference
\	Quotes the following character
'	Quotes a string, preventing all substitution
"	Quotes a string, allowing only variable and command substitution
` ... `	Performs command substitution
[ ]	Character class in an ambiguous file reference
$	References a variable

. (dot builtin)	Executes a command
#	Begins a comment
{ }	Surrounds the contents of a function
: (null builtin)	Returns *true*
&& (Boolean AND)	Executes command on right only if command on left succeeds
|| (Boolean OR)	Executes command on right only if command on left fails
! (Boolean NOT)	Reverses exit status of a command
$() (not in tcsh)	Performs command substitution
[ ]	Evaluates an arithmetic expression

Γ. Εισαγωγή Δεδομένων σε Scripts

Μπορούμε να περάσουμε δεδομένα σε ένα script με τη βοήθεια της εντολής read. Το ακόλουθο παράδειγμα χρησιμοποιεί την εντολή read ώστε να διαβάσει το όνομα, την ηλικία και το φύλο του χρήστη τα οποία και τυπώνει στην οθόνη στη συνέχεια.

```
#!/bin/bash
clear
echo "Please enter your name"
read name
echo "Please enter your age"
read age
echo "Please enter your sex. Male/Female"
read sex
echo "So you're a $age year old $sex called $name"
```

Για το ίδιο παράδειγμα, μπορούμε να υιοθετήσουμε μια εκδοχή της read ώστε να φτιάξουμε scripts που χαρακτηρίζονται από περισσότερη εποπτικότητα.

```
#!/bin/bash
clear
read -p "Please enter your name  : " name
echo ""
read -p "Please enter your age  : " age
echo ""
read -p "Please enter your sex. Male/Female  : " sex
echo ""
echo "So you're a $age year old $sex called $name"
```

Το ακόλουθο script διαβάζει μια πρόταση από το πληκτρολόγιο και στη συνέχεια εμφανίζει την πρόταση με κεφαλαίους χαρακτήρες.

```
#!/bin/bash
read -p "Please give a sentence : " foo
var=$(echo $foo | tr "{a-z}" "{A-Z}")
```

```
# {a-z} Matches a through z
# {A-Z} matches A through Z
echo $var
```

Το ακόλουθο script κρυπτογραφεί ένα αρχείο το όνομα του οποίου δίνεται ως παράμετρος από το πληκτρολόγιο με τη βοήθεια του εργαλείου gpg.

```
#!/bin/bash
echo "Welcome, I am ready to encrypt a file/folder for you"
echo "currently I have a limitation, Place me to the same folder, where a file to be
encrypted is present"
echo "Enter the Exact File Name with extension"
read file;
gpg -c $file
echo "I have encrypted the file successfully..."
echo "Now I will be removing the original file"
rm -rf $file
```

Το παραπάνω script σβήνει το αρχικό αρχείο. Η αποκρυπτογράφηση γίνεται με την ακόλουθη εντολή.

gpg -d filename.gpg > filename

Δ. Παράμετροι Γραμμής Εντολής

Μέσα από τα scripts έχουμε τη δυνατότητα να προσπελάσουμε τις παραμέτρους που θέτονται από τον χρήστη στη γραμμή εντολής (positional parameters). Αυτό γίνεται μέσω των αναφορών $0, $1, $2, … για την πρώτη, δεύτερη, τρίτη, κ.ο.κ. παράμετρο. Η $0 παράμετρος αντιστοιχεί πάντα στην εντολή – script ενώ η έκφραση $# μας δίνει των πλήθος των παραμέτρων.

Για την κατανόηση της χρήσης των παραμέτρων της γραμμής εντολής, εκτελέστε τα ακόλουθα scripts.

```
#!/bin/bash
# posit-param: script to view command line parameters
echo ""
Number of arguments: $#
\$0 = $0
\$1 = $1
\$2 = $2
\$3 = $3
\$4 = $4
\$5 = $5
\$6 = $6
\$7 = $7
\$8 = $8
\$9 = $9
```

Σημείωση. Για να καθορίσουμε περισσότερες από 9 παραμέτρους χρησιμοποιούμε την έκφραση ${10}, ${55}, ${211}, κ.λπ.

```
#!/bin/bash
foo=$1+$2
echo $foo
echo "Positional Parameters"
echo '$0 = ' $0
echo '$1 = ' $1
echo '$2 = ' $2
```

Στο δεύτερο παράδειγμα, το script εκτελεί το άθροισμα των δύο πρώτων παραμέτρων της γραμμής εντολής.

```
#!/bin/bash
foo=`expr $1 + $2`
echo $foo
echo "Positional Parameters"
echo '$0 = ' $0
echo '$1 = ' $1
echo '$2 = ' $2
```

Ε. Εντολή if

Η σύνταξη της εντολής if έχει ως εξής:

if commands; **then**
commands
[**elif** commands; **then**
commands...]
[**else**
commands]
fi

όπου commands είναι μια λίστα από οποιασδήποτε εντολές. Για την κατανόηση της εντολής if θα εκτελέσουμε τα ακόλουθα παραδείγματα:

Script 1

```
echo -n "word 1: "
read word1
echo -n "word 2: "
read word2
if test "$word1" = "$word2"; then
        echo "Match"
fi
echo "End of program."
```

Script 2

```
if test $# -eq 0; then
        echo "You must supply at least one argument."
        exit 1
fi
echo "Program running."
```

Script 3

```
if test $# -eq 0; then
        echo "You must supply at least one argument."
        exit 1
fi
if test -f "$1"; then
        echo "$1 is an ordinary file in the working directory"
else
        echo "$1 is NOT an ordinary file in the working directory"
fi
```

Script 4

```
echo -n "word 1: "
read word1
echo -n "word 2: "
read word2
echo -n "word 3: "
read word3
if [ "$word1" = "$word2" -a "$word2" = "$word3" ]; then
        echo "Match: words 1, 2, & 3"
elif [ "$word1" = "$word2" ]; then
        echo "Match: words 1 & 2"
elif [ "$word1" = "$word3" ]; then
        echo "Match: words 1 & 3"
elif [ "$word2" = "$word3" ]; then
        echo "Match: words 2 & 3"
else
        echo "No match"
fi
```

Z. Κωδικοί Εξόδου

Όλες οι εντολές του Linux κατά τη διαδικασία εκτέλεσής τους επιστρέφουν ένα κωδικό εξόδου που είναι ένας ακέραιος από το 0 μέχρι το 255. Αυτός ο αριθμός ονομάζεται κωδικός εξόδου (exit status). Ο αριθμός 0 αντιπροσωπεύει την επιτυχή εκτέλεση της εντολής ενώ οποιαδήποτε άλλη τιμή αντιπροσωπεύει την αποτυχία.

Για να δείτε τον κωδικό εξόδου μιας εντολής, εκτελέστε μια εντολή ls και αμέσως μετά δώστε την εντολή echo $?. Δοκιμάστε επίσης να δώσετε τις ακόλουθες εντολές:

```
[kk@linux ~]$ true
[kk@linux ~]$ echo $?
0
[kk@linux ~]$ false
[kk@linux ~]$ echo $?
1
[kk@linux ~]$ if true; then echo "It's true."; fi
It's true.
[kk@linux ~]$ if false; then echo "It's true."; fi
```

[kk@linux ~]$

Η. Εντολή test

Η εντολή test χρησιμοποιείται τις περισσότερες φορές με την εντολή if ώστε να εκτελέσει πλήθος ελέγχων και συγκρίσεων. Η σύνταξη της εντολής είναι η ακόλουθη:

test expression

ή

[expression **]**

με το expression να είναι μια συνθήκη που αποτιμάται σε αληθής ή ψευδής.

Ο ακόλουθος πίνακας παρουσιάζει τις παραμέτρους της εντολής που αφορούν σε αρχεία.

Expression	Is true if . . .
file1 -ef *file2*	*file1* and *file2* have the same inode numbers (the two filenames refer to the same file by hard linking).
file1 -nt *file2*	*file1* is newer than *file2*.
file1 -ot *file2*	*file1* is older than *file2*.
-b *file*	*file* exists and is a block-special (device) file.
-c *file*	*file* exists and is a character-special (device) file.
-d *file*	*file* exists and is a directory.
-e *file*	*file* exists.
-f *file*	*file* exists and is a regular file.
-g *file*	*file* exists and is set-group-ID.
-G *file*	*file* exists and is owned by the effective group ID.
-k *file*	*file* exists and has its "sticky bit" set.
-L *file*	*file* exists and is a symbolic link.
-O *file*	*file* exists and is owned by the effective user ID.
-p *file*	*file* exists and is a named pipe.
-r *file*	*file* exists and is readable (has readable permission for the effective user).
-s *file*	*file* exists and has a length greater than zero.
-S *file*	*file* exists and is a network socket.
-t *fd*	*fd* is a file descriptor directed to/from the terminal. This can be used to determine whether standard input/output/error is being redirected.
-u *file*	*file* exists and is setuid.
-w *file*	*file* exists and is writable (has write permission for the effective user).
-x *file*	*file* exists and is executable (has execute/search permission for the effective user).

Το επόμενο παράδειγμα τυπώνει το είδος ενός αρχείου της δομής καταλόγων / αρχείων:

Script 5

```
#!/bin/bash
```

```
# test-file: Evaluate the status of a file
FILE=~/.bashrc
if [ -e "$FILE" ]; then
        if [ -f "$FILE" ]; then
                echo "$FILE is a regular file."
        fi
        if [ -d "$FILE" ]; then
                echo "$FILE is a directory."
        fi
        if [ -r "$FILE" ]; then
                echo "$FILE is readable."
        fi
        if [ -w "$FILE" ]; then
                echo "$FILE is writable."
        fi
        if [ -x "$FILE" ]; then
                echo "$FILE is executable/searchable."
        fi
else
        echo "$FILE does not exist"
        exit 1
fi
exit
```

Οι ακόλουθοι πίνακες παραθέτουν πιθανές επιλογές κατά τη χρήση της εντολής test.

Αλφαριθμητικά

Expression	Is true if . . .
string	*string* is not null.
-n *string*	The length of *string* is greater than zero.
-z *string*	The length of *string* is zero.
string1 = *string2* *string1* == *string2*	*string1* and *string2* are equal. Single or double equal signs may be used, but the use of double equal signs is greatly preferred.
string1 != *string2*	*string1* and *string2* are not equal.
string1 > *string2*	*string1* sorts after *string2*.
string1 < *string2*	*string1* sorts before *string2*.

Παράδειγμα:

Script 6

```
#!/bin/bash
# test-string: evaluate the value of a string
ANSWER=maybe
if [ -z "$ANSWER" ]; then
        echo "There is no answer."
```

```
        exit 1
fi
if [ "$ANSWER" = "yes" ]; then
        echo "The answer is YES."
elif [ "$ANSWER" = "no" ]; then
        echo "The answer is NO."
elif [ "$ANSWER" = "maybe" ]; then
        echo "The answer is MAYBE."
else
        echo "The answer is UNKNOWN."
fi
```

Ακέραιοι

Expression	Is true if . . .
integer1 -eq *integer2*	*integer1* is equal to *integer2*.
integer1 -ne *integer2*	*integer1* is not equal to *integer2*.
integer1 -le *integer2*	*integer1* is less than or equal to *integer2*.
integer1 -lt *integer2*	*integer1* is less than *integer2*.
integer1 -ge *integer2*	*integer1* is greater than or equal to *integer2*.
integer1 -gt *integer2*	*integer1* is greater than *integer2*.

Παράδειγμα:

Script 7

```
#!/bin/bash
# test-integer: evaluate the value of an integer.
INT=-5
if [ -z "$INT" ]; then
        echo "INT is empty."
        exit 1
fi
if [ $INT -eq 0 ]; then
        echo "INT is zero."
else
        if [ $INT -lt 0 ]; then
                echo "INT is negative."
        else
                echo "INT is positive."
        fi
        if [ $((INT % 2)) -eq 0 ]; then
                echo "INT is even."
        else
                echo "INT is odd."
        fi
fi
```

Λογικοί Τελεστές

Operation	test	[[ ]] and (( ))
AND	-a	&&
OR	-o	||
NOT	!	!

Εφαρμογή: Δημιουργία Μενού Επιλογών

Script 8

```
#!/bin/bash
# read-menu: a menu driven system information program
clear
echo "
Please Select:
1. Display System Information
2. Display Disk Space
3. Display Home Space Utilization
0. Quit
"
read -p "Enter selection [0-3] > "
if [[ $REPLY =~ ^[0-3]$ ]]; then
        if [[ $REPLY == 0 ]]; then
                echo "Program terminated."
                exit
        fi
        if [[ $REPLY == 1 ]]; then
                echo "Hostname: $HOSTNAME"
                uptime
                exit
        fi
        if [[ $REPLY == 2 ]]; then
                df -h
                exit
        fi
        if [[ $REPLY == 3 ]]; then
                if [[ $(id -u) -eq 0 ]]; then
                        echo "Home Space Utilization (All Users)"
                        du -sh /home/*
                else
                        echo "Home Space Utilization ($USER)"
                        du -sh $HOME
                fi
                exit
        fi
else
        echo "Invalid entry."
        exit 1
```

fi

Σημείωση: Κανονικές Εκφράσεις (Regular Expressions)

Στις κανονικές εκφράσεις μπορούμε να χρησιμοποιήσουμε ένα σύνολο μετα-χαρακτήρων ώστε να παράξουμε πολύπλοκα 'ταιριάσματα' στις διάφορες εντολές που εκτελούμε στο Linux. Οι μετα-χαρακτήρες αυτοί είναι:

- ^ : ταιριάζει με την αρχή μιας γραμμής
- $: ταιριάζει με το τέλος μιας γραμμής
- . : ταιριάζει με οποιδήποτε χαρακτήρα
- [ ] : ταιριάζει με οποιοδήποτε χαρακτήρα ανάμεσα στα σύμβολα
- {N} : ο προηγούμενος χαρακτήρας(ες) ταιριάζει ακριβώς N φορές
- {N, } : ο προηγούμενος χαρακτήρας(ες) ταιριάζει N ή περισσότερες φορές
- {N,M} : ο προηγούμενος χαρακτήρας(ες) ταιριάζει N ή περισσότερες φορές αλλά το πολύ M φορές
- ? : ο προηγούμενος(οι) χαρακτήρας(ες) ταιριάζουν το πολύ μια φορά
- * : μηδέν η περισσότερες εμφανίσεις
- + : ταριάζει με μια η περισσότερες εμφανίσεις του προηγούμενου χαρακτήρα
- \> : ταιριάζει με το τέλος μιας λέξης
- \< : ταιριάζει με την αρχή μιας γραμμής
- \ : αναιρεί τη σημασία ενός μετα-χαρακτήρα
- | : χαρακτήρας σωλήνωσης

Ο τελεστής σύγκρισης για κανονικές εκφράσεις είναι ο =~ .

Παραδείγματα:

```
if [[ $digit =~ [0-9] ]]; then
   echo "$digit is a digit"
else
   echo "oops"
fi
```

```
echo -n "Your answer> "
read REPLY
if [[ $REPLY =~ ^[0-9]+$ ]]; then
   echo Numeric
else
   echo Non-numeric
fi
```