Πανεπιστήμιο Δυτικής Μακεδονίας
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα

Ενότητα 4: Σενάρια Φλοιού Unix. Βρόχοι Επανάληψης, αλλαγή ροής εκτέλεσης. Αριθμητικές πράξεις, χειρισμός αλφαριθμητικών και συνθήκες.

Επιβλέπων: Δασυγένης Μηνάς
Παυλίδου Ελένη

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Λειτουργικών Συστημάτων
http://arch.ece.uowm.gr/courses/os/

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Βρόχος επανάληψης for (1/4)

- Σε πολλά προγράμματα υπάρχει η αναγκαιότητα επαναληπτικής εκτέλεσης κάποιων εντολών μέχρι να επιτευχθεί το επιθυμητό αποτέλεσμα.
  Στα προγράμματα κελύφους μία από τις δομές επαναλήψεων που προσφέρονται είναι ο βρόχος for που συντάσσεται ως :

```
for <μεταβλητή>   in <λίστα τιμών>
 do
  εντολή 1
  εντολή 2
  ……
  εντολή n
done
```

Βρόχος επανάληψης for (2/4)

- Η <μεταβλητή> μπορεί να είναι οποιαδήποτε, σύμφωνα με τους κανόνες ονοματολογίας.
  Η <λίστα τιμών> πρέπει να είναι μία λίστα με τις διακριτές τιμές που θέλουμε να πάρει η <μεταβλητή> κατά την εκτέλεση του βρόχου, χωριζόμενες με κενό.
  Όσες τιμές έχει μέσα η <λίστα τιμών> τόσες φορές θα εκτελεστεί και ο βρόχος `for`.

Βρόχος επανάληψης for (3/4)

- Παράδειγμα:

```
for i in 1 2 3 4 5 6 7 8 9 10 ;

    do
       echo $i

    done
```

Ο παραπάνω βρόχος εμφανίζει στην οθόνη τους αριθμούς από 1 έως 10, έναν σε κάθε γραμμή.

Βρόχος επανάληψης for (4/4)

- **Χρήση εισαγωγικών**

Τα απλά εισαγωγικά (‘single quotes’) άρουν την μεταχαρακτηρική ιδιότητα όλων των συμβόλων εκτός από τον εαυτό τους.

Τα διπλά εισαγωγικά (“double quotes”) άρουν την μεταχαρακτηρική ιδιότητα όλων των συμβόλων εκτός από τον εαυτό τους, την ανάποδη κάθετο (backslash \ ) και το δολάριο ($). Τα ανάποδα εισαγωγικά (` back quotes `) προκαλούν την εκτέλεση της εντολής που περικλείουν.

Βρόχος επανάληψης while (1/2)

- Η εντολή `while` χρησιμοποιείται όπως σε όλες τις γλώσσες για την εκτέλεση ενός κύκλου εντολών κάτω από κάποια συνθήκη (Conditional loop). Η σύνταξή της είναι ως εξής:

```
while <συνθήκη> ;
  do
     ……
    <εντολές>
     ……
done
```

Βρόχος επανάληψης while (2/2)

- Η <συνθήκη> συντάσσεται με παρόμοιο τρόπο με τις συνθήκες `if`. Στην θέση της συνθήκης μπαίνει κάποια εντολή.

- Ο βρόχος `while` εκτελείται όσο η συνθήκη είναι αληθής. Όσο δηλαδή η εντολή που παίζει το ρόλο της συνθήκης επιστρέφει κωδικό σωστής εκτέλεσης (κωδικός 0).

Η εντολή expr (1/3)

- Χρήση της `expr` για αριθμητικές πράξεις.

i=`expr $i +1`	Προσθέτει το 1 στο i και βάζει το αποτέλεσμα ξανά στο i
var=`expr $var – 5`	Αφαιρεί το 5 από το var και βάζει το αποτέλεσμα ξανά στο var
var1=`expr $var \* 3`	Πολλαπλασιάζει το var με το 3 και βάζει το αποτέλεσμα στο var1
var2=`expr $var \/ 4`	Διαιρεί το var με το 4 και βάζει το αποτέλεσμα στο var2

Η εντολή expr (2/3)

- Προσοχή όλη η έκφραση περικλείεται σε ανάποδα εισαγωγικά (`) που σημαίνουν ότι η εντολή εκτελείται. Η χρήση του \ πριν από το * και το / είναι αναγκαία γιατί αυτά τα σύμβολα για το κέλυφος είναι μεταχαρακτήρες.

- Παράδειγμα `expr` σε `while loop`.

```
i=1
while [ $i -lt 100 ];
do
 echo $i; i=`expr $i + 1`
done
```

Η εντολή expr (3/3)

- Η εντολή `expr` χρησιμοποιείται για να υπολογίζει αριθμητικές εκφράσεις ακεραίων. Ως γνωστόν το κέλυφος του UNIX δεν αναγνωρίζει αριθμούς, οπότε τον ρόλο αυτό τον αναλαμβάνει η `expr`.

- Η χρήση του `i` είναι ως μετρητής εκτελέσεων του βρόχου.

- Η `expr` επιστρέφει κωδικό εκτέλεσης μεγαλύτερο ή ίσο του 2 όταν προσπαθούμε να κάνουμε πράξεις με μεταβλητές που ΔΕΝ είναι αριθμοί.

Εντολές break και continue (1/4)

- Η εντολή **`break`** χρησιμοποιείται για την έξοδο από τον τρέχοντα βρόχο πριν την κανονική έξοδο. Χρησιμοποιείται όταν δεν γνωρίζουμε εκ των προτέρων πόσες φορές θα εκτελεστεί ο βρόχος, για παράδειγμα όταν η έξοδος από το βρόχο εξαρτάται από τα δεδομένα εισόδου.

- Προσοχή, η **`break`** βγαίνει από το βρόχο, όχι από το σενάριο. Αυτό μπορεί να δειχτεί αν βάλουμε μια εντολή **`echo`** στο τέλος του σεναρίου. Αυτή η **`echo`** θα εκτελεστεί μετά τη **`break`** (όταν ο χρήστης εισάγει "**0**"). Σε ένθετους βρόχους, η **`break`** επιτρέπει το καθορισμό του βρόχου από τον οποίο βγαίνουμε.

Εντολές break και continue (2/4)

- Η εντολή **`continue`** συνεχίζει την εκτέλεση ενός βρόχου **`for`**, **`while`**, **`until`** ή **`select`** από το επόμενο βήμα.
  Όταν χρησιμοποιείται σε βρόχο **`for`**, η μεταβλητή ελέγχου λαμβάνει την τιμή του επομένου στοιχείου της λίστας.
  Όταν όμως χρησιμοποιείται με δομή `while` ή `until`, η εκτέλεση συνεχίζεται με την **TESTCOMMAND** στην αρχή του βρόχου.

Εντολές break και continue (3/4)

- Στο ακόλουθο παράδειγμα, τα ονόματα αρχείων μετατρέπονται όλα ώστε να περιέχουν μόνο πεζά γράμματα. Αν δεν απαιτείται μετατροπή, μια δήλωση `continue` συνεχίζει την εκτέλεση του βρόχου από την αρχή.
  Η δομή `continue` είναι χρήσιμη αν οι εργασίες είναι βαριές και πιθανώς η εκτέλεσή τους να μην είναι απαραίτητη, αν και εφόσον εισάγουμε τους κατάλληλους ελέγχους στα σωστά σημεία του σεναρίου μας, γλιτώνοντας έτσι την κατάχρηση πόρων του συστήματος.

Εντολές break και continue (4/4)

```
[carol@octarine ~/test] cat tolower.sh
#!/bin/bash
# This script converts all file names
containing upper case characters into file#
names containing only lower cases.
LIST="$(ls)"
for name in "$LIST"; do
if [[ "$name" != *[[:upper:]]* ]]; then
continue
fi
ORIG="$name"
NEW=`echo $name | tr 'A-Z' 'a-z'`
mv "$ORIG" "$NEW"
echo "new name for $ORIG is $NEW"
done
```

Η εντολή case (1/2)

- Οι ένθετες δηλώσεις `if` μπορεί να είναι καλές, αλλά όσο μεγαλώνει ο αριθμός των επιλογών, γίνονται μάλλον περίπλοκες. Για πολλαπλές επιλογές χρησιμοποιούμε την `case` η οποία έχει την παρακάτω σύνταξη:

```
case EXPRESSION in
CASE1) COMMAND-LIST;;
CASE2) COMMAND-LIST;;
......
CASEN) COMMAND-LIST;;
esac
```

Η εντολή case (2/2)

- Κάθε περίπτωση ορίζεται από μια έκφραση προς ταύτιση. Εκτελούνται οι εντολές στο COMMAND-LIST που ακολουθεί την πρώτη επιτυχημένη ταύτιση. Το σύμβολο “|” (οριζόντια κάθετος) χρησιμοποιείται για το χωρισμό πολλαπλών εκφράσεων και ο τελεστής “`)`” τερματίζει τη λίστα των εκφράσεων.
  Κάθε περίπτωση μαζί με τις εκφράσεις και τις εντολές της λέγεται μια πρόταση (clause). Κάθε πρόταση πρέπει να τελειώνει με “`;;`”. Κάθε δήλωση `case` τερματίζεται με τη δήλωση `esac`.

Η εντολή cut (1/5)

- Όταν θέλουμε να εμφανίσουμε ένα τμήμα ενός αλφαριθμητικού, τότε μπορούμε να χρησιμοποιήσουμε το εξωτερικό πρόγραμμα `cut`.

- `cut -f 2,4 file`
  Αποκοπή των στηλών 2 και 4, ενός αρχείου λίστας (για τη μετάβαση από τη μία στήλη στην άλλη, πληκτρολογούμε το χαρακτήρα TAB).

Η εντολή cut (2/5)

- `cut –f 2-4 file`
  Αποκοπή των στηλών 2 έως 4, ενός αρχείου λίστας.

- `cut –c 6,9 file`
  Αποκοπή των χαρακτήρων 6 και 9 (όλων των γραμμών), ενός αρχείου λίστας.

- `cut –c 6-9 file`
  Αποκοπή των χαρακτήρων 6 έως 9 (όλων των γραμμών), ενός αρχείου λίστας.

Η εντολή cut (3/5)

- Για ένα πρώτο παράδειγμα του `cut`, θα χρησιμοποιήσουμε το αρχείο `/etc/passwd`. Χρησιμοποιούμε αυτό το αρχείο επειδή τα πεδία του αρχείου διαχωρίζονται από τον χαρακτήρα "`:`", πράγμα το οποίο κάνει πολύ εύκολη την εργασία του.

- `cut -f1 -d: /etc/passwd`

  Η παραπάνω εντολή `cut` μας λέει τα ακόλουθα:

Η εντολή cut (4/5)

- Εκτυπώστε το πρώτο πεδίο (`-f1`).
- Τα πεδία οριοθετούνται από τον χαρακτήρα `":"` `(-d :)`.
- Χρησιμοποιήστε το αρχείο `/etc/passwd` ως είσοδο.

➢ Μπορούμε να χρησιμοποιήσουμε την εντολή `cut` με την εντολή `ls` για να πάρουμε μια λίστα με ονόματα αρχείων στον τρέχοντα κατάλογο. Το `cut` δέχεται δεδομένα με τη χρήση της διασύνδεσης των εντολών δια μέσου της κάθετης γραμμής `|`.

Η εντολή cut (5/5)

- Παράδειγμα: `ls -al | cut-c44-`

❖ Παρατήρηση! Σε αυτή τη γραμμή χρησιμοποιείται η εντολή `cut` "`-c`". Αυτή η εντολή μας λέει το εξής:

- Εκτελέστε την εντολή "`ls -al`".
- Μετακινήστε την έξοδο της εντολής στην εντολή `cut`.
- Η εντολή `cut` εκτυπώνει τα πάντα από κάθε γραμμή ξεκινώντας από τη στήλη 44 μέσω του άκρου της γραμμής (-c44-).

Ανακατεύθυνση Εξόδου (1/2)

- Υπάρχουν δύο χειριστές για την ανακατεύθυνση της εξόδου μιας εντολής σε ένα αρχείο αντί της οθόνης.

- Το σύμβολο **>** δημιουργεί ένα νέο αρχείο αν δεν υπάρχει ή αντικαθιστά το αρχείο εάν υπάρχει ήδη.
- Το **>>** δημιουργεί επίσης ένα νέο αρχείο αν δεν υπάρχει, αλλά προσαρτά το κείμενο στο τέλος του αρχείου αν υπάρχει ήδη, αντί να αντικαταστήσει το αρχείο.

- Προσοχή! Όταν ανακατευθύνουμε μια έξοδο σεναρίου κελύφους σε ένα αρχείο, δεν θα εμφανίζεται στην κονσόλα.

Ανακατεύθυνση Εξόδου (2/2)

- Για να ανακατευθύνουμε την έξοδο μιας εντολής σε ένα αρχείο, καθορίζουμε το χειριστή **>** ή το **>>** και μετά δίνουμε τη διαδρομή σε ένα αρχείο στο οποίο θέλουμε να ανακατευθυνθεί η έξοδος.
  Για παράδειγμα, η εντολή `ls` απαριθμεί τα αρχεία και τους φακέλους στον τρέχοντα κατάλογο.

- Παράδειγμα: `ls -l /bin > lsfile.txt`

  Θα τοποθετηθεί η έξοδος μέσα στο αρχείο `lsfile.txt` και έτσι δε θα εμφανιστεί τίποτα στην οθόνη.

Η εντολή exit

- Ένας χρήστης μπορεί να έχει πολλά κελύφη τα οποία να τρέχουν το ένα πάνω στο άλλο και όλα αυτά να τρέχουν και εφαρμογές. Για να κλείσουμε ένα κέλυφος και να βρεθούμε στο προηγούμενο εκτελούμε την εντολή **`exit`**.
  Αν εκτελέσουμε αυτήν την εντολή ενώ βρισκόμαστε στο κατώτατο-πρώτο κέλυφος τότε θα βγούμε από το σύστημα.
  Άλλη εντολή που μας βγάζει από το σύστημα είναι η εντολή `logout`. Αυτή η εντολή στα περισσότερα συστήματα μπορεί να εκτελεστεί μόνο εφόσον βρισκόμαστε στο κατώτατο κέλυφος.

Λίγη ακόμα if (1/2)

- Μέσα σε μια εντολή `if` μπορούμε να χρησιμοποιήσουμε έναν έλεγχο αρχείου.

- Παράδειγμα:

```
if [ -f $HOME/lib/functions ];
then
cat $HOME/lib/functions | grep myfunction
fi
```

- Μερικοί έλεγχοι που σχετίζονται με αρχεία είναι:
  - `-d` FILE, αληθής αν το αρχείο FILE είναι κατάλογος.
  - `-e` FILE, αληθής αν το αρχείο FILE υπάρχει.
  - `-f` FILE, αληθής αν το αρχείο FILE υπάρχει και είναι κανονικό αρχείο.

Λίγη ακόμα if (2/2)

- Εκτός από τους ελέγχους αρχείων, υπάρχουν και οι έλεγχοι συμβολοσειρών. Μερικοί έλεγχοι που σχετίζονται με συμβολοσειρές είναι:

  - `-z STRING`, αληθής αν η συμβολοσειρά `STRING` είναι άδεια.
  - `-n STRING`, αληθής αν η συμβολοσειρά `STRING` δεν είναι άδεια.

Εισαγωγή στο πρόγραμμα grep (1/2)

- Η εντολή `grep` αναζητά σε αρχεία κειμένου γραμμές που περιέχουν ένα πρότυπο. Όταν βρίσκει μια ταύτιση, αντιγράφει τη συγκεκριμένη γραμμή στη τυπική έξοδο (ως προεπιλογή) ή σε όποια άλλη έξοδο έχουμε ορίσει με τη χρήση επιλογών.

- Αν και η `grep` υποθέτει ταύτιση σε κείμενο, δεν έχει συγκεκριμένο όριο στο μέγεθος γραμμής, πέρα από τη διαθέσιμη μνήμη, και μπορεί να ταυτίσει αυθαιρέτους χαρακτήρες μέσα σε μια γραμμή. Αν το τελικό byte της γραμμής εισόδου δεν είναι newline, η `grep` σιωπηρά προσθέτει ένα. Αφού η αλλαγή γραμμής είναι και διαχωριστικό προτύπων, δεν υπάρχει τρόπος να ταυτίσουμε χαρακτήρες αλλαγής γραμμής.

Εισαγωγή στο πρόγραμμα grep (2/2)

- Ένα απλό παράδειγμα χρήσης του `grep`:

```
grep apple fruitlist.txt
```

Το `grep` θα επέστρεφε σε αυτή την περίπτωση όλες τις γραμμές του αρχείου `fruitlist.txt` που θα περιείχαν τουλάχιστον μια εμφάνιση της λέξης 'apple'. Δεν θα επέστρεφε σαν έξοδο γραμμές που περιέχουν τη λέξη 'Apple' (κεφαλαίο Α) γιατί η προκαθορισμένη λειτουργία του είναι ευαίσθητη στα κεφαλαία-μικρά. Όμως, όπως οι περισσότερες εντολές του Unix, μπορεί να δεχθεί σημαίες που μπορούν να αλλάξουν αυτή τη λειτουργία.

Τέλος Ενότητας

Πανεπιστήμιο Δυτικής Μακεδονίας