Πανεπιστήμιο Δυτικής Μακεδονίας
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα

Ενότητα 12:Ακολουθίες ANSI και συναρτήσεις σε σενάρια φλοιού. Χειρισμός ώρας και καταλόγων.

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.

Χαρακτήρας διαφυγής (1/6)

- Στην επιστήμη των υπολογιστών και στις τηλεπικοινωνίες καλούμε **χαρακτήρα διαφυγής** έναν μόνο χαρακτήρα ο οποίος σε μια ακολουθία χαρακτήρων σηματοδοτεί ότι οι χαρακτήρες που ακολουθούν λαμβάνουν διαφορετική ερμηνεία. Ο όρος **ακολουθία διαφυγής** αναφέρεται σε έναν χαρακτήρα διαφυγής και στους ακόλουθους χαρακτήρες ή χαρακτήρα που αλλάζει η ερμηνεία τους.

Χαρακτήρας διαφυγής (2/6)

- Η ακολουθία ANSI ξεκινάει πάντα από ένα χαρακτήρα, που ονομάζεται χαρακτήρας απόδρασης.
  Τον χαρακτήρας ESC με δεκαδικό κώδικα 27
  και δεκαεξαδικό κώδικα 1B.
  Δε γράφουμε τους τρεις χαρακτήρες ESC, αλλά χρησιμοποιούμε τον ένα χαρακτήρα που δημιουργείται από το 0x1B.

Χαρακτήρας διαφυγής (3/6)

- Τα τερματικά **ANSI / VT100** και οι εξομοιωτές τερματικών δεν είναι μόνο σε θέση να εμφανίζουν ασπρόμαυρο κείμενο. Μπορούν να εμφανίσουν χρώματα και μορφοποιημένα κείμενα χάρη στις ακολουθίες διαφυγής. Αυτές οι ακολουθίες αποτελούνται από τον προαναφερόμενο χαρακτήρα **Escape** (που συχνά αντιπροσωπεύεται από " `^[` " ή " `<Esc>` ") ακολουθούμενοι από κάποιους άλλους χαρακτήρες:
" `<Esc>[` *`FormatCode`* `m` ".

Χαρακτήρας διαφυγής (4/6)

- Στο Bash, ο χαρακτήρας `<Esc>` μπορεί να ληφθεί με τις ακόλουθες προσθήκες:

- `\e`
- `\033`
- `\x1B`

- Ακολουθούν παραδείγματα χρήσης.

Χαρακτήρας διαφυγής (5/6)

Κωδικός (Bash)	Προεπισκόπηση
`echo -e "\ e [31mHello World \ e [0μ"`	Hello World
`echo -e "\ 033 [31mHello \ e [0μ Κόσμος"`	Hello World

Χαρακτήρας διαφυγής (6/6)

- Η επιλογή -e της εντολής `echo` επιτρέπει την ανάλυση των ακολουθιών διαφυγής.
- Η ακολουθία " `\e[0m` " αφαιρεί όλα τα χαρακτηριστικά (μορφοποίηση και χρώματα). Μπορεί να είναι καλή ιδέα να το προσθέσουμε στο τέλος κάθε έγχρωμου κειμένου.
- Τα παραδείγματα εδώ βρίσκονται στο **Bash** αλλά οι ακολουθίες διαφυγής **ANSI / VT100** μπορούν να χρησιμοποιηθούν σε όλες τις γλώσσες προγραμματισμού.

Οι ακολουθίες ANSI (1/3)

- Αν χρησιμοποιηθούν οι ακολουθίες διαφυγής ANSI (ANSI escape code), τα συμβατά με αυτές τερματικά μπορούν να δείξουν κείμενο με χρώμα:

```
FGRED=`echo "\033[31m"`
FGCYAN=`echo "\033[36m"`
BGRED=`echo "\033[41m"`
FGBLUE=`echo "\033[35m"`
BGGREEN=`echo "\033[42m"`

NORMAL=`echo "\033[m"`
```

Οι ακολουθίες ANSI (2/3)

Εν συνεχεία της προηγούμενης διαφάνειας:

```
echo "${FGBLUE} Text in blue ${NORMAL}"
echo "Text normal"
echo "${BGRED} Background in red"
echo "${BGGREEN} Background in Green and
   back to Normal ${NORMAL}"
```

- Ακολουθεί πίνακας με παραδείγματα ειδικών ακολουθιών ANSI μορφοποίησης κειμένου, χρώματος και άλλων λειτουργιών που δέχεται το τερματικό.

Οι ακολουθίες ANSI (3/3)

Escape sequence	Meaning
`ESC[nA`	Move up *n* lines
`ESC[nB`	Move down *n* lines
`ESC[nC`	Move right *n* lines
`ESC[nD`	Move left *n* lines
`ESC[m;nH`	Move cursor to (*m,n*)
`ESC[sJ`	Clear screen from cursor (0 to end, 1 from start, 2 all)
`ESC[sK`	Clear line from cursor (0 to end, 1 from start, 2 all)
`ESC[nL`	Insert *n* lines at cursor
`ESC[nM`	Delete *n* lines from cursor
`ESC[nP`	Delete *n* chars from cursor
`ESC[n@`	Insert *n* chars at cursor
`ESC[nm`	Enable rendition *n* (0=normal, 4=bold, 5=blinking, 7=reverse)
`ESC M`	Scroll the screen backward if the cursor is on the top line

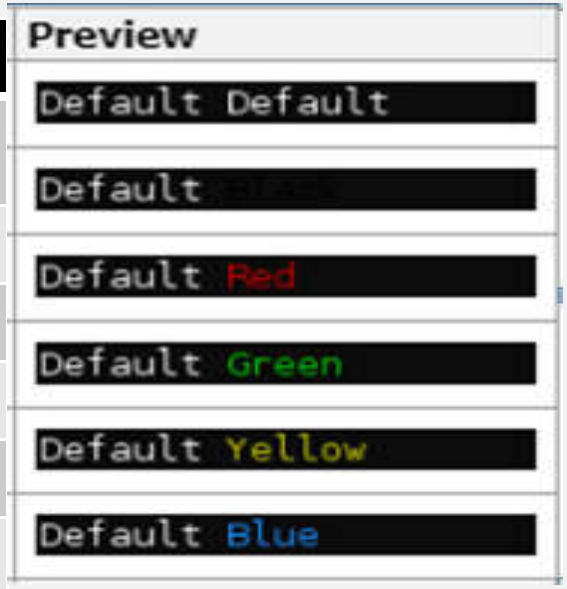
Παραδείγματα (1/2)

- Ακολουθούν οι πιο συχνά υποστηριζόμενες ακολουθίες ελέγχου για τη μορφοποίηση κειμένου.

Code	Description	Example
1	Bold/Bright	`echo -e "Normal \e[1mBold"`
2	Dim	`echo -e "Normal \e[2mDim"`
4	Underlined	`echo -e "Normal \e[4mUnderlined"`
5	Blink	`echo -e "Normal \e[5mBlink"`
7	Reverse (invert the foreground and background colors)	`echo -e "Normal \e[7minverdet"`
8	Hidden (useful for passwords)	`echo -e "Normal \e[8mHidden"`

Παραδείγματα (2/2)

Code	Color	Example
39	Default foreground color	`echo -e "Default \e[39mDefault"`
30	Black	`echo -e "Default \e[30mBlack"`
31	Red	`echo -e "Default \e[31mRed"`
32	Green	`echo -e "Default \e[32mGreen"`
33	Yellow	`echo -e "Default \e[33mYellow"`
34	Blue	`echo -e "Default \e[34mBlue"`



Πρόγραμμα tput() (1/4)

- Οι διακριτές λειτουργίες που υποστηρίζει το **`tput`** είναι οι Cursor Attributes και Text Attributes. (Από τη σελίδα του tput).

- Το βοηθητικό πρόγραμμα **`tput`** χρησιμοποιεί τη βάση δεδομένων **`terminfo`** για να κάνει τις τιμές εξαρτημένων από το τερματικό δυνατότητες και πληροφορίες διαθέσιμες στο κέλυφος, για να προετοιμάσει ή να επαναφέρει το τερματικό ή να επιστρέψει το `longname` του τύπου τερματικού που ζητήθηκε.

Πρόγραμμα tput() (2/4)

- Το αποτέλεσμα εξαρτάται από τον τύπο των μεταβλητών.

- `string`: Το `tput` γράφει τη συμβολοσειρά στην τυπική έξοδο. Δεν παρέχεται νέα γραμμή.

- `integer`: Το `tput` γράφει τη δεκαδική τιμή στην τυπική έξοδο, με μια νέα γραμμή που βρίσκεται στο τέλος.

- `boolean`: Το `tput` θέτει απλώς τον κωδικό εξόδου (**0** για TRUE, **1** για FALSE) και δεν εμφανίζει τίποτα στην τυπική έξοδο.

Πρόγραμμα tput() (3/4)

- Πριν χρησιμοποιήσουμε μια τιμή που επιστρέφεται στην τυπική έξοδο, η εφαρμογή πρέπει να ελέγξει τον κωδικό εξόδου (π.χ. **$?**) για να βεβαιωθούμε ότι είναι **0.**

- `Ttype`: υποδεικνύει τον *τύπο* του τερματικού. Κανονικά, αυτή η επιλογή είναι περιττή, επειδή η προεπιλογή λαμβάνεται από τη μεταβλητή περιβάλλοντος **TERM** . Εάν καθορίζεται το **`-T`**, τότε οι μεταβλητές κελύφους **LINES** και **COLUMNS** θα αγνοηθούν και το λειτουργικό σύστημα δεν θα ερωτηθεί για το πραγματικό μέγεθος οθόνης.

Πρόγραμμα tput() (4/4)

- Τον αριθμό των στηλών του τερματικού μας παραθύρου τον εκτυπώνουν οι παράμετροι:
  `tput_cols`.

- Ενώ τον αριθμό των γραμμών του τερματικού μας παραθύρου τον εκτυπώνουν οι παράμετροι:
  `tput_lines`.

Χειρισμός ώρας (1/3)

```
char *ctime(const time_t *time)
char *asctime(const tm *time)
```

- Οι παραπάνω συναρτήσεις μεταφράζουν το χρόνο από τύπο `time_t` ή δομή `tm` στη γνωστή αναγνώσιμη συμβολοσειρά που εμφανίζει η εντολή `date` του UNIX/Linux.

Χειρισμός ώρας (2/3)

```
time_t time(time_t *tloc)
```

- Επιστρέφει τον ημερολογιακό χρόνο από την 00:00:00 GMT, Jan. 1, 1970 (γνωστή ως 'the epoch', GMT=Greenwitch Mean Time), μετρούμενο σε δευτερόλεπτα.
- Αν ο δείκτης `tloc` δεν είναι NULL, η τιμή που επιστρέφει η συνάρτηση επίσης αποθηκεύεται στη θέση που δείχνει ο `tloc`.

Χειρισμός ώρας (3/3)

- Σε περίπτωση επιτυχίας η συνάρτηση `time()` επιστρέφει το χρόνο, ενώ σε περίπτωση αποτυχίας -1. Ο τύπος `time_t` ορίζεται ως `long int` στη βιβλιοθήκη `time.h`.

- Εκτός από την απλή δομή που ορίζει το χρόνο σε δευτερόλεπτα υπάρχουν πολλές διαφορετικές δομές χρόνου ανάλογα με την χρήση, π.χ. τοπικός ημερολογιακός χρόνος (ημέρα, μήνας, θερινή ώρα, ζώνη ώρας κλπ) ή ώρα μεγάλης ακρίβειας, (σε microsec ή σε nanosec).

Sizeof()

- Το `sizeof()` (το οποίο δεν είναι συνάρτηση αλλά ΤΕΛΕΣΤΗΣ της C), επιστρέφει το μέγεθος σε BYTES είτε ενός τύπου είτε μίας μεταβλητής:

```
main()
{
    char c;
    short s;
    int i;
    double d;
        int a[10];
        printf("%d\n", sizeof(c));
        printf("%d\n", sizeof(s));
        printf("%d\n", sizeof(char));
}
```

Χειρισμός Καταλόγων (1/6)

- Συνάρτηση **`opendir`**.

`opendir(όνομα φακέλου)`

- Σαν όρισμα βάζουμε απλά το όνομα του φακέλου αν είναι μέσα στον ίδιο φάκελο με το αρχείο μας.
- Αλλιώς βάζουμε το path (σχετικό ή απόλυτο).

Χειρισμός Καταλόγων (2/6)

- Το αποτέλεσμα:

- Επιστρέφει ένα αντικείμενο τύπου φάκελο αρχείων.
- Επιστρέφει `false` αν δεν υπάρχει ο φάκελος.
- Χρησιμοποιείται από άλλες συναρτήσεις, που θα δούμε παρακάτω.

Χειρισμός Καταλόγων (3/6)

- Συνάρτηση **`readdir()`:**

- Δέχεται σαν όρισμα το αντικείμενο που επιστρέφει η `opendir()`.

- Το αποτέλεσμα:

- Επιστρέφει το επόμενο αρχείο μέσα στο φάκελο που επιλέξαμε.

- Επιστρέφει `false` αν αποτύχει να επιστρέψει αρχείο.

Χειρισμός Καταλόγων (4/6)

- Συνάρτηση **`closedir`**`()`.

- Δέχεται σαν όρισμα το αντικείμενο που επιστρέφει η `opendir()`.

- Το αποτέλεσμα:

- Κλείνει το φάκελο που επιλέξαμε με την `opendir()` όταν τελειώσουμε τις εργασίες μας με αυτόν.

- Επιστρέφει `false` αν αποτύχει να κλείσει τον φάκελο.

Χειρισμός Καταλόγων (5/6)

- Συνάρτηση `rewinddir`:

- Επιστρέφει στην αρχή ενός καταλόγου για να τον ξαναδιαβάσει.

- Άνοιγμα (`opendir`).
- Κλείσιμο (`closedir`).
- Αρχή (`rewinddir`).

Χειρισμός Καταλόγων (6/6)

```
int stat(char *path, struct stat *buf)
```

- Η **`stat()`** παρέχει πληροφορίες για το αρχείο που το όνομά του δίνεται από το path.
  Δεν απαιτούνται δικαιώματα ανάγνωσης, εγγραφής ή εκτέλεσης στο αρχείο αλλά όλοι οι κατάλογοι στη διαδρομή θα πρέπει να επιτρέπουν ανάγνωση.

Το πρόγραμμα valgrind

- Η βασική ιδέα πίσω από την αρχιτεκτονική του Valgrind είναι η διαίρεση μεταξύ των “core” και “tools” του. Ο πυρήνας παρέχει την κοινή υποδομή χαμηλού επιπέδου που υποστηρίζει τα όργανα του προγράμματος, συμπεριλαμβανομένου του μεταγλωττιστή J86 x86-to-x86 JIT, του διαχειριστή μνήμης χαμηλού επιπέδου, του χειρισμού σήματος και ενός χρονοπρογραμματισμού (για pthreads). Παρέχει επίσης ορισμένες υπηρεσίες που είναι χρήσιμες σε μερικά αλλά όχι όλα τα εργαλεία, όπως υποστήριξη για καταγραφή και καταστολή σφαλμάτων.

Τέλος Ενότητας

Πανεπιστήμιο Δυτικής Μακεδονίας