ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Λειτουργικά Συστήματα

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No:12

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

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 του Πανεπιστημίου 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.

Περιεχόμενα

1. Σκοπός της άσκησης

- Ακολουθίες ANSI σε σενάρια φλοιού *(tput, printf)* και σε προγράμματα C.
- Συναρτήσεις σε σενάρια φλοιού.
- Χειρισμός ώρας *(ctime(), time(), sizeof()).*
- Χειρισμός Καταλόγων *(opendir(), readdir(), closedir(), rewinddir(), stat()).*
- Βελτιστοποίηση προγραμμάτων μέσω του εργαλείου valgrind.

2. Παραδοτέα

(A) 16 ερωτήσεις
(C) 8 ασκήσεις
(B) 2 bonus

ΠΡΟΣΟΧΗ: **Το 2015 το FreeBSD αποφάσισε να χρησιμοποιεί και να υποστηρίζει τον μεταγλωττιστή `clang` σε αντίθεση με το linux που χρησιμοποιεί το `gcc`. Σε περίπτωση που εργάζεστε σε FreeBSD μηχάνημα (όπως το zafora) θα πρέπει να αντικαταστήστε τις εντολές `gcc` με τις εντολές `clang` (οι παράμετροι παραμένουν ίδιοι). Σε περίπτωση που εργάζεστε σε Linux μηχάνημα (όπως το pleiades), δε χρειάζεται κάποια αλλαγή.**

3. Διαχείριση Τερματικού

Τα τερματικά στο UNIX υποστηρίζουν κάποιες ειδικές ακολουθίες *(ονομάζονται ακολουθίες ANSI)*, που επιτρέπουν την τοποθέτηση του δρομέα σε οποιοδήποτε σημείο της οθόνης, ή την χρήση χρωμάτων. Με αυτόν τον τρόπο μπορούμε να δημιουργήσουμε εφαρμογές πλήρους οθόνης τερματικού ή να υλοποιήσουμε πιο πολύπλοκες λειτουργίες εμφάνισης.

4. Οι ακολουθίες ANSI

Είναι κωδικοποιημένες σειρές από χαρακτήρες ελέγχου, και ορίζονται από διάφορα πρότυπα *(π.χ. ANSI X3.41-1974).* Οι ακολουθίες αυτές συνδέονται με συγκεκριμένο τύπο τερματικού, συνήθως vt100, όμως συνήθως λόγω της συμβατότητας, λειτουργούν ικανοποιητικά και σε διαφορετικά τερματικά *(π.χ. xterm)*. Η ακολουθία ANSI ξεκινάει πάντα από ένα χαρακτήρα, που ονομάζεται χαρακτήρας απόδρασης *(escape character)* ο οποίος είναι το ESC με δεκαεξαδικό τιμή 0x1B. Δε γράφουμε τους τρεις χαρακτήρες ESC, αλλά χρησιμοποιούμε τον ένα χαρακτήρα που δημιουργείται από το 0x1B. Επειδή κάποιες φορές είναι δύσκολο να γράψουμε αυτό το χαρακτήρα, υπάρχουν κάποιες εναλλακτικές μορφές που εξαρτώνται από το

τερματικό. Μετά τον χαρακτήρα απόδρασης, ακολουθεί το string ελέγχου και μηδέν ή περισσότεροι χαρακτήρες παράμετροι, αναλόγως του τι θέλουμε να επιτύχουμε. Μπορείτε να δείτε όλο το πρότυπο ANSI στο αρχείο ansicode.txt . Οι χαρακτήρες ελέγχου είναι πάρα πολλοί. Σε αυτή την άσκηση θα χρησιμοποιήσουμε κάποιους πολύ βασικούς χαρακτήρες. Οι ANSI ακολουθίες χρησιμοποιούνται και στο bash shell και σε προγράμματα σε C.

5. ANSI ακολουθίες στο BASH SHELL

Στο φλοιό BASH οι ακολουθίες ANSI χρησιμοποιούν τον χαρακτήρα απόδρασης **\033** ή **\E** και στη συνέχεια υπάρχει μια αγκύλη **[** . Μετά ακολουθεί η υπόλοιπη ακολουθία. Η εντολή echo που θα χρησιμοποιηθεί για να θέσει το χρώμα θα πρέπει να έχει μια συγκεκριμένη παράμετρο.

Ψάξτε στη σελίδα βοήθειας για το bash, και συγκεκριμένα στην παράγραφο για το echo, και σημειώστε την παράμετρο που ενεργοποιεί τη μετάφραση των ακολουθιών (`"interpretation of the following backslash-escaped characters is enabled"`).

_____ **(A1)**

Αυτή την παράμετρο θα την τοποθετείτε κάθε φορά που χρησιμοποιείτε μια ακολουθία ANSI (π.χ. `echo -παραμετρος "ANSI ESCAPE SEQUENCE"`).

Τα βασικά χρώματα που υποστηρίζονται στο BASH δίνονται παρακάτω:

Παράμετρος	Επεξήγηση
`3 0 m`	Set **foreground** to color #0 - black
`3 1 m`	Set **foreground** to color #1 - red
`3 2 m`	Set **foreground** to color #2 - green
`3 3 m`	Set **foreground** to color #3 - yellow
`3 4 m`	Set **foreground** to color #4 - blue
`3 5 m`	Set **foreground** to color #5 - magenta
`3 6 m`	Set **foreground** to color #6 - cyan
`3 7 m`	Set **foreground** to color #7 - white
`3 9 m`	Set **default** color as foreground color

Τα οποία μπορούν να συνδυαστούν με τις ιδιότητες των χαρακτήρων:

Παράμετρος	Επεξήγηση
`0 m`	Reset all attributes
`1 m`	Set "bright" attribute
`2 m`	Set "dim" attribute
`4 m`	Set "underscore" *(underlined text)* attribute
`5 m`	Set "blink" attribute
`7 m`	Set "reverse" attribute

Παράμετρος	Επεξήγηση
8 m	Set "hidden" attribute

Έτσι μπορούμε να δημιουργήσουμε πολλούς συνδυασμούς, μερικούς από τους οποίους παραθέτονται παρακάτω:

```
Black          0;30    Dark Gray      1;30
Blue           0;34    Light Blue     1;34
Green          0;32    Light Green    1;32
Cyan           0;36    Light Cyan     1;36
Red            0;31    Light Red      1;31
Purple         0;35    Light Purple   1;35
Brown          0;33    Yellow         1;33
Light Gray     0;37    White          1;37
```

Επίσης, αν θέλετε να χρησιμοποιήσετε διαφορετικό background, θα τοποθετήσετε μπροστά από τους παραπάνω χαρακτήρες, ένα νούμερο, από το 40 έως το 49, π.χ. 44 για μπλε υπόβαθρο, 41 για κόκκινο υπόβαθρο, όπως φαίνεται στην παρακάτω λίστα:

Παράμετρος	Επεξήγηση
40 m	Set **foreground** to color #0 - black
41 m	Set **foreground** to color #1 - red
42 m	Set **foreground** to color #2 - green
43 m	Set **foreground** to color #3 - yellow
44 m	Set **foreground** to color #4 - blue
45 m	Set **foreground** to color #5 - magenta
46 m	Set **foreground** to color #6 - cyan
47 m	Set **foreground** to color #7 - white
49 m	Set **default** color as foreground color

Οι παραπάνω κώδικες μπορεί να συνδυάζονται μεταξύ τους, είτε να είναι ξεχωριστοί.

Για παράδειγμα, η ακολουθία **44;1;31** τοποθετεί το light red πάνω σε μπλε υπόβαθρο. Η ακολουθία ολοκληρώνεται με το γράμμα m.

Π.χ. (**"\033[44;1;31m"**). Η ίδια ακολουθία μπορεί να γραφεί και ως:

"\033[44m \033[1m \033[31m"

5.1 Ερωτήσεις Α2 έως Α8

Επιβεβαιώστε ότι χρησιμοποιείτε το φλοιό BASH ή SH (και όχι csh / tcsh / zsh). Δώστε την εντολή ή εντολές που χρησιμοποιήσατε για να το επιβεβαιώστε: _____ **(A2)**

Αν δε χρησιμοποιείτε το BASH, τότε θα πρέπει να δώστε **bash** για να ανοίξετε μια νέα συνεδρία με το συγκεκριμένο φλοιό.

Δώστε την εντολή echo για να τροποποιήσετε το χρώμα του τερματικού σε πράσινο. Θα διαπιστώσετε ότι από εδώ και πέρα όλοι οι χαρακτήρες θα είναι πράσινοι. _____ **(A3)**

Σε περίπτωση που θέλουμε να επαναφέρουμε τις αρχικές ρυθμίσεις θα στείλουμε την ακολουθία **0m** . Στην ίδια γραμμή μπορούμε να έχουμε πολλές ακολουθίες.
π.χ. **`echo "ANSI ESCAPE SEQ" κείμενο "ANSI ESCAPE SEQ"`**

Δώστε μια εντολή echo στην οποία ταυτόχρονα θέτουμε το χρώμα των χαρακτήρων σε μπλε, εκτυπώνουμε το όνομά μας, και στη συνέχεια επαναφέρουμε τις αρχικές ρυθμίσεις. _____ **(A4)**

Εκτός από την echo, μπορούμε να χρησιμοποιήσουμε μια εντολή του λειτουργικού συστήματος που εκτυπώνει το κείμενο μαζί με τη μορφοποίηση. Η εντολή είναι η printf[1].

Δώστε **`man 1 printf`** για να βρείτε τη σελίδα βοήθειας του προγράμματος. Αν δίνατε **`man 3 printf`** θα σας εμφανίζονταν πάλι μια σελίδα βοήθειας.
Σε τι διαφέρει η **`man 1...`** με τη **`man 3`** …;
_____ **(A5)**

Το πρόγραμμα printf χρησιμοποιείται κυρίως με δυο τρόπους. Στον πρώτο τρόπο τοποθετούμε κατευθείαν μια παράμετρο, το string προς εκτύπωση.
Π.χ. **`printf 'STRING'`**

Στο δεύτερο τρόπο τοποθετούμε δυο ή παραπάνω παραμέτρους. Η πρώτη παράμετρος δίνει το string μορφοποίησης, και οι επόμενες τα στοιχεία που θα εκτυπωθούν μορφοποιημένα, με αντιστοίχηση 1-προς-1. Προσέξτε ότι αντί για %s όταν θέλετε να χρησιμοποιήσετε ANSI μορφοποίηση, θα έχετε %b
`printf "%s %10d %b\n" "Hello" 1 "\033[0m"`
{στο παράδειγμα, το %s αντιστοιχεί στο πρώτο string Hello, το %10d αντιστοιχεί στην επόμενη μεταβλητή, και το %b είναι η μορφοποίηση για την ακολουθία ANSI} .

Δώστε μια εντολή printf στην οποία ταυτόχρονα θέτουμε το χρώμα των χαρακτήρων σε μπλε, εκτυπώνουμε το όνομά μας, και στη συνέχεια επαναφέρουμε τις αρχικές ρυθμίσεις._____ **(A6)**

Εκτός από τις παραπάνω ακολουθίες ANSI για χρήση χρωμάτων, μπορούμε να χρησιμοποιήσουμε ακολουθίες ANSI για μετακίνηση χαρακτήρων σε οποιοδήποτε σημείο της οθόνης. Αυτό μπορεί να γίνει είτε με το echo είτε με το printf. Όμως, σε κάθε περίπτωση αν μετακινηθεί ο χαρακτήρας **ΔΕΝ πρέπει να δοθεί εντολή αλλαγής γραμμής**, γιατί ο δρομέας θα επανέλθει στην επόμενη γραμμή στην αρχή.

[1] Η printf μπορεί να χρησιμοποιηθεί με την ίδια μορφή και στο csh/tcsh για την εκτύπωση ANSI ακολουθιών, κάτι που δε μπορεί να το κάνει η echo.

Για αυτό το λόγο η εντολή echo θα πρέπει να χρησιμοποιηθεί με την παράμετρο **-n** *(echo -n)* ενώ η printf δε θα πρέπει να έχει \n.

Για να μετακινήσουμε ένα cursor σε ένα οποιοδήποτε σημείο, χρησιμοποιούνται οι παρακάτω ANSI ακολουθίες:

ANSI ακολουθίες	Επεξήγηση
\033[<L>;<C>f	Μετακινείται ο δρομέας στη γραμμή L στήλη C
\033[<N>A	Μετακινείται ο δρομέας N γραμμές προς τα πάνω (above)
\033[<N>B	Μετακινείται ο δρομέας N γραμμές προς τα κάτω (bellow)
\033[<N>C	Μετακινείται ο δρομέας N στήλες εμπρόσθια (δεξιά)
\033[<N>D	Μετακινείται ο δρομέας N στήλες πρόσθια (αριστερά)
\033[2J	Καθαρισμός της οθόνης και μετακίνηση δρομέα στο (0,0)
\033[K	Διαγραφή μέχρι το τέλος της γραμμής (Kill Line)
\033[s	Αποθήκευση τρέχουσας τοποθεσίας δρομέα *(αν υποστηρίζει το τερματικό)*
\033[u	Επαναφορά τρέχουσας τοποθεσίας δρομέα *(αν υποστηρίζει το τερματικό)*

Ένα χρήσιμο πρόγραμμα για τις ακολουθίες ANSI στο τερματικό είναι το **tput**. Βρείτε από τη σελίδα του tput τις 2 διακριτές λειτουργίες που υποστηρίζει:

_____ **(A7)**

Το **tput** μπορεί να μας εκτυπώσει τον αριθμό των στηλών και των γραμμών του παραθύρου του τερματικού μας. Αυτοί οι 2 αριθμοί είναι πολύ σημαντικοί, γιατί μπορούμε να τους χρησιμοποιήσουμε για την τοποθέτηση μηνυμάτων σε οποιοδήποτε σημείο της οθόνης.

Από τη σελίδα βοήθειας του **tput** σε συνδυασμό με τα attributes ή ονόματα variables που βρίσκονται στο terminfo (**man terminfo**) βρείτε τις 2 παραμέτρους που εκτυπώνουν τον αριθμό των στηλών του τερματικού μας παραθύρου *(Print the number of columns for the current terminal)* και εκτυπώνουν τον αριθμό των γραμμών του τερματικού μας παραθύρου:

_____**(A8)**

Αυτές οι δυο μεταβλητές χρησιμοποιούνται συνήθως ως εξής: Στην αρχή του script εκτελούμε το tput και αναθέτουμε σε δικές μας μεταβλητές τις τιμές που επιστρέφονται. Π.χ.

```
num_lines=`tput XXX`
num_cols=`tput YYY`
```

όπου **XXX** και **YYY** είναι οι παράμετροι που έχετε βρει στην προηγούμενη ερώτηση. Αν θέλετε να εκτυπώσετε κάτι στο κέντρο, τότε θα χρησιμοποιήσετε τις συντεταγμένες **num_lines / 2** και **num_cols / 2** . Μάλιστα προκειμένου να βρίσκεται το κείμενο ακριβώς στο κέντρο του τερματικού, θα πρέπει να ξεκινήσετε την εκτύπωση λίγο πιο πριν κατά ένα offset ίσο με το μισό του μήκους string. Δηλαδή, αν το string είναι “Hello World from ANSI”, το οποίο έχει 21 χαρακτήρες μήκος, θα πρέπει να εκτυπωθεί στη συντεταγμένη **num_lines /2** και **num_cols / 2 + 21 / 2** . Σε

περίπτωση που θέλετε να κεντράρετε ένα πλαίσιο από πολλές γραμμές, θα πρέπει ομοίως να υπολογίσετε και το κατακόρυφο offset που πρέπει να αφαιρέσετε από το num_lines.

5.2 Το παραδοτέο C1

(C1) Να δημιουργήσετε σενάριο φλοιού σε sh στο οποίο θα ζητείται από το χρήστη να πληκτρολογήσει ένα όνομα, και στη συνέχεια αυτό θα εκτυπώνεται στο κέντρο του τερματικού του με τις ακολουθίες ANSI. Μπορείτε να χρησιμοποιήσετε τις παρακάτω οδηγίες:

- Βρίσκετε τον αριθμό των γραμμών του τερματικού και τον τοποθετείτε σε μια μεταβλητή.
- Βρίσκετε τον αριθμό των στηλών του τερματικού και τον τοποθετείτε σε μια μεταβλητή.
- Εκτυπώνετε την ερώτηση.
- Διαβάζετε την είσοδο από το πληκτρολόγιο *(read)*
- Χρησιμοποιώντας το echo και τη μεταβλητή του read από πριν, που διασωληνώνεται στο πρόγραμμα **`wc`** με την παράμετρο **`-c`** *(υπολογισμός χαρακτήρων)* υπολογίζετε το μέγεθος του string, και το τοποθετείτε σε μια μεταβλητή.
- Υπολογίζετε το offset με το πρόγραμμα **`expr`**, το οποίο είναι η προηγούμενη μεταβλητή διαιρεμένη με το 2 , και το αναθέτετε σε μια μεταβλητή.
- Υπολογίζετε το οριζόντιο κέντρο που είναι ο αριθμός των γραμμών διαιρεμένος με το 2. Τοποθετείται σε μια μεταβλητή y_center.
- Υπολογίζετε το κατακόρυφο κέντρο που είναι ο αριθμός των στηλών διαιρεμένος με το 2, και στη συνέχεια αφαιρείται το offset. Τοποθετείται σε μια μεταβλητή x_center.
- Χρησιμοποιείτε το printf για να εκτυπώσετε στις συντεταγμένες x_center , y_center προσέχοντας ότι όταν έχετε ANSI ακολουθία με μεταβλητές, πρέπει να χρησιμοποιείτε διπλά εισαγωγικά για να γίνεται variable expansion.
- Στο τέλος επαναφέρετε το τερματικό με την ANSI ακολουθία `"\033[0m"`.

6. Συναρτήσεις στο φλοιό

Μια από τις δυνατότητες του φλοιού SH είναι η χρήση συναρτήσεων, δηλαδή κομματιών κώδικα που μπορούμε να τα επαναχρησιμοποιήσουμε. Η συνάρτηση στο sh δηλώνεται πάντα πριν τη χρησιμοποιήσουμε *(συνήθως στις πρώτες γραμμές)*, ως εξής:

```
myfunction()
{
….
return val
```

```
}
```

Η συνάρτηση αυτή επιστρέφει στο πρόγραμμα με την τιμή επιστροφής val.

Η συνάρτηση καλείται μέσα στο script ως:

```
Myfunction
```

Επίσης, η συνάρτηση αυτή μπορεί να δεχτεί και παραμέτρους,

π.χ. `myfunction 10 100  myname "hello world"`

Σε αυτή την περίπτωση, αυτόματα μέσα στη συνάρτηση δημιουργούνται οι μεταβλητές $1 έως $9 που αντιστοιχούν στις παραμέτρους 1-προς-1. Στο παράδειγμά μας η $1 είναι το 10, $2 είναι το 100, $3 είναι το myname και $4 είναι το "hello world".

Οι έμπειροι προγραμματιστές, τοποθετούν όλες τις συναρτήσεις σε ένα διαφορετικό αρχείο *(π.χ. functions.sh)*, το οποίο το ενσωματώνουν στον κωδικά τους με τη χρήση της τελείας **.** , στην αρχή του κώδικα ως εξής:

```
#!/bin/sh
. ./functions.sh
..υπόλοιπος κώδικας
```

(C2) Να δημιουργήσετε σενάριο φλοιού σε sh, στο οποίο θα ζητείται από το χρήστη να πληκτρολογήσει ένα όνομα, και στη συνέχεια αυτό θα εκτυπώνεται στο κέντρο του τερματικού του με τις ακολουθίες ANSI. Το σενάριο θα χρησιμοποιεί μια συνάρτηση σε ξεχωριστό αρχείο με το όνομα **functions.sh** , η οποία θα δέχεται 3 παραμέτρους . Η πρώτη παράμετρος θα είναι το num_lines *(αριθμός γραμμών του τερματικού)*, η δεύτερη παράμετρος θα είναι το num_cols *(αριθμός στηλών του τερματικού)*, και η τρίτη παράμετρος το string προς εκτύπωση μέσα σε διπλά εισαγωγικά. Η συνάρτηση θα εκτυπώνει το string στο κέντρο των παραπάνω συντεταγμένων, λαμβάνοντας υπόψιν το offset.

(C3) Να δημιουργήσετε σενάριο φλοιού σε sh, στο οποίο θα υλοποιείτε μια **progress bar**. Συγκεκριμένα, θα εκτυπώνετε αρχικά χωρίς αλλαγή γραμμής, το [.....] *(10 τελείες)* και στη συνέχεια θα χρησιμοποιείτε την ακολουθία ANSI για μετακίνηση Ν στηλών προς τα αριστερά *(στην πρώτη τελεία)*. Μετά θα δημιουργείτε ένα while βρόχο επανάληψης για τις 10 τελείες, στις οποίες θα εκτυπώνετε με κόκκινο χρώμα το χαρακτήρα αστέρι * , πάνω στις τελείες, και θα ακολουθεί μια παύση sleep για 1 sec. Στο τέλος, θα κάνετε αλλαγή γραμμής, και θα επαναφέρετε τις κανονικές ρυθμίσεις στο τερματικό. Μπορείτε να δείτε το video **ProgressBar_inShell.avi** που εμφανίζει το αποτέλεσμα που θέλουμε να πετύχουμε.

(BONUS1 +50%) Να δημιουργήσετε σενάριο φλοιού σε sh το οποίο θα μετακινείται ένα αστέρι από άκρη σε άκρη. Συγκεκριμένα, θα εκτυπώσετε 10 τελείες μέσα σε αγκύλες *(όπως προηγουμένως)*, και στη συνέχεια θα εκτυπώσετε ένα αστέρι τέρμα αριστερά, το οποίο κάθε sec θα προχωράει μια θέση *(δηλαδή θα τοποθετείται τελεία*

στη θέση που ήταν) προς τα δεξιά. Όταν φτάσει στην άλλη άκρη του συνόλου μας, θα αρχίσει να μετακινείται προς τα αριστερά. Αυτό θα πρέπει να γίνει για 5 πλήρεις κύκλους, και στη συνέχεια θα σταματήσει η εκτέλεση του script.

7. ANSI ακολουθίες σε C προγράμματα

Δεν υπάρχει κάποια διαφορά ως προς τη χρήση των ANSI ακολουθιών σε προγράμματα C. Ο χαρακτήρας απόδρασης είναι ο \e , και μπορεί να χρησιμοποιηθεί σε οποιοδήποτε πρόγραμμα εκτυπώνει στην οθόνη. Για παράδειγμα, το παρακάτω πρόγραμμα:

```
#include <stdio.h>
main()
{
printf("\e[31mHello\e[32m,\e[34mBlue\e[32mWorld\e[0m.\n")
;
}
```

θα εκτυπώσει στην οθόνη: 

Σε αυτό το σημείο θα πρέπει να θυμηθείτε ότι όταν στην `printf()` δεν τοποθετούμε χαρακτήρα αλλαγής γραμμής `\n`, θα πρέπει να τοποθετήσουμε μια `fflush(NULL)` αμέσως μετά, για να σταλεί η γραμμή στο τερματικό προς εκτύπωση *(η συνάρτηση printf() έχει "**line buffered**" έξοδο)*.

(C4) Να δημιουργήσετε πρόγραμμα σε C, στο οποίο θα υλοποιεί μια progress bar, σύμφωνα με τις οδηγίες που είχαν δοθεί σε προηγούμενη άσκηση.

Η δυνατότητα της χρήσης των ακολουθιών ANSI στο C κώδικα, του δίνει νέες δυνατότητες, όπως π.χ. η δημιουργία εφαρμογών πλήρους οθόνης τερματικού. Επίσης, μπορούμε να εκτυπώσουμε χαρακτήρες ελέγχου ASCII, όπως ο χαρακτήρας "\007" που είναι ο χαρακτήρας BELL *(δηλαδή, ο χαρακτήρας που όταν εκτυπωθεί παράγει ένα beep)*.

Σε περίπτωση που θέλουμε να βρούμε τις γραμμές και στήλες του τερματικού, θα χρησιμοποιήσουμε την κλήση συστήματος ioctl, όπως παρουσιάζει το παρακάτω κομμάτι κώδικα:

```
#include <sys/ioctl.h>
#include <stdio.h>

int main()
{
        struct winsize w;
        ioctl(0, TIOCGWINSZ, &w);
```

```
        printf("lines %d\n", w.ws_row);
        printf("columns %d\n", w.ws_col);
        return 0;
}
```

8. Χειρισμός ώρας στο UNIX

Το ΛΣ UNIX έχει πολλές συναρτήσεις για την εμφάνιση πληροφοριών χρόνου. Μια από αυτές τις συναρτήσεις είναι η **`ctime()`** η οποία δέχεται μια παράμετρο. Η παράμετρος αυτή είναι ο αριθμός των δευτερολέπτων από τη χρονική στιγμή που ονομάζεται epoch, και είναι η 1η Ιανουαρίου 1970. Προκειμένου να βρούμε πόσα δευτερόλεπτα έχουν περάσει από την epoch, ως την τρέχουσα χρονική στιγμή του συστήματος μας, θα χρησιμοποιήσουμε τη συνάρτηση **`time()`**. Η **`time()`** επιστρέφει μια τιμή τύπου time_t και καλείται με παράμετρο NULL. Στη συνέχεια καλείται η **`ctime()`** με μια παράμετρο, η οποία είναι η τιμή που έχει επιστρέψει η **`time()`**. Θα πρέπει να δοθεί προσοχή σε κάποιες παραμέτρους που είναι pointers.

(C5) Να δημιουργήσετε πρόγραμμα σε C, στο οποίο θα εκτυπώνεται η τρέχουσα ώρα. Επίσης, θα εκτυπώνεται το μέγεθος σε Bytes της μεταβλητής time_t.

- Αρχικά θα καλείται η **`time()`** με παράμετρο NULL και θα ακολουθεί η κλήση της **`ctime()`**. Στη συνέχεια θα εκτυπώνεται το μήνυμα "`current time is`" και θα ακολουθεί αυτό που έχει επιστρέψει η **`ctime()`**.
- Για να βρεθεί το μέγεθος σε Bytes, θα γίνει η κλήση της συνάρτησης **`sizeof()`**.

(BONUS2 +50%) Να δημιουργήσετε πρόγραμμα σε C, το οποίο θα υλοποιεί screensaver του τερματικού εμφανίζοντας την ώρα. Συγκεκριμένα:

- θα καθαρίζει αρχικά την οθόνη, με την κατάλληλη ακολουθία ASCII.
- Θα βρίσκει τις γραμμές και στήλες του τερματικού.
- Θα υπολογίζει μια random τιμή γραμμής από 0 έως τον αριθμό των γραμμών.
- Θα υπολογίζει μια random τιμή στήλης από 0 έως τον αριθμό των στηλών αφαιρούμενο το 20 *(για να χωρέσει η ώρα σε μια γραμμή).*
- Θα εκτυπώνει σε αυτό το σημείο την ώρα.
- Θα περιμένει 5 sec.
- Η διαδικασία θα επαναλαμβάνεται από το σημείο καθαρισμού της οθόνης.
- Επίσης, θα δημιουργήσετε ένα signal handler, ο οποίος θα αλλάζει το χρώμα των γραμμάτων σε μια νέα random τιμή, κάθε φορά που θα στέλνετε το σήμα SIGUSR1.

9. Χειρισμός Καταλόγων στο UNIX

Εκτός από το χειρισμό των αρχείων που είχαμε δει σε προηγούμενο εργαστήριο, υπάρχουν και κλήσεις συστήματος για το χειρισμό καταλόγων.

Η κλήση συστήματος **`opendir()`** πόσες παραμέτρους δέχεται, και τι τιμή επιστρέφει; _____ **(A9)**

Η κλήση συστήματος **`readdir()`** πόσες παραμέτρους δέχεται, και τι τιμή επιστρέφει; _____ **(A10)**

Η κλήση συστήματος **`closedir()`** πόσες παραμέτρους δέχεται, και τι τιμή επιστρέφει; _____ **(A11)**

Να σημειώσετε ότι ενώ η **`opendir()`** καλείται μια φορά, η **`readdir()`** κάθε φορά που καλείται επιστρέφει την επόμενη καταχώρηση στον κατάλογο, μέχρι να επιστρέψει NULL που σημαίνει ότι δεν υπάρχουν άλλες καταχωρήσεις. Επίσης υπάρχει η συνάρτηση **`rewinddir()`** που επιστρέφει το δρομέα της επόμενης καταχώρησης καταλόγου στην αρχή.

Η δομή dirent ορίζει τα πεδία που αντιστοιχούν σε κάθε αρχείο, και μπορεί να βρεθεί από το **`man dirent`** . Το πεδίο d_fileno αντιστοιχεί στο i-node, το d_type στο είδος της καταχώρησης *(π.χ. αρχείο ή κατάλογος ή συμβολικός δεσμός)* και το d_name στο όνομα της καταχώρησης.

(C6) Να δημιουργήσετε πρόγραμμα σε C, το οποίο θα εκτυπώνει τις καταχωρήσεις *(όνομα, είδος, και αριθμό i-node)* που βρίσκονται στον τρέχων κατάλογο. Μπορείτε να χρησιμοποιήσετε τις παρακάτω οδηγίες:

- Κάντε include τα αρχεία που πρέπει, όπως μπορείτε να δείτε από τα manpages των κλήσεων συστήματος που θα χρησιμοποιήσουμε.
- Ορίζετε ένα δείκτη καταλόγου mydir, ως DIR *mydir;
- Ορίζετε μια δομή για τις καταχωρήσεις, τύπου dirent *(struct dirent *mydirent)*
- Κάνετε κλήση της **`opendir()`** με παράμετρο “.” για τον τρέχων κατάλογο.
- Δημιουργείτε ένα ατέρμονο βρόχο *(while(1))* στον οποίο καλείτε την **`readdir()`** με παράμετρο το δείκτη που έχει επιστρέψει η **`opendir()`**, ελέγχετε αν είναι NULL, οπότε και κάνετε break ή διαφορετικά, εκτυπώνετε μορφοποιημένα 3 στήλες: Η πρώτη στήλη έχει μήκος 20 χαρακτήρων και φέρει το όνομα *(mydirent->d_name),* η επόμενη έχει μήκος 5 χαρακτήρων και φέρει την τιμή του τύπου της καταχώρησης, και η τελευταία έχει μήκος 10 χαρακτήρων και έχει το i_node της καταχώρησης.
- Στο τέλος θα καλείτε το **`closedir()`**.

(C7) Να δημιουργήσετε πρόγραμμα σε C, το οποίο θα εκτυπώνει τις καταχωρήσεις *(όνομα, είδος, και αριθμό i-node).* Τροποποιήστε το προηγούμενο αρχείο ώστε:

- Το πρόγραμμα θα δέχεται μια παράμετρο από τη γραμμή εντολής *(δηλαδή θα χρησιμοποιήσετε* `main(int argc,char *argv[])` *)* . Αν δεν έχει δοθεί μια παράμετρος, θα ερωτάται ο χρήστης να δώσει το όνομα του καταλόγου που θέλει να δει τα αρχεία.

- Ο κατάλογος που θα ανοίγει δε θα είναι ο **“.”** , αλλά ο κατάλογος που έχει δώσει ο χρήστης.

- Θα υπάρχει έλεγχος για το αν μπορεί να ανοίξει ο κατάλογος *(έλεγχος της τιμής επιστροφής).* Αν δε μπορεί να ανοίξει, θα ενημερώνεται κατάλληλα ο χρήστης.

- Θα εκτυπώνονται οι καταχωρήσεις. Αν είναι αρχεία θα εκτυπώνονται με πράσινο χρώμα, αν είναι κατάλογοι με κόκκινο.

Μια πολύ χρήσιμη συνάρτηση, είναι και η `stat()` η οποία επιστρέφει πληροφορίες για ένα αρχείο, ακριβώς όπως ακριβώς το πρόγραμμα stat.

Προκειμένου να βρούμε τη σελίδα βοήθειας *(man page)* για την κλήση συστήματος `stat()`, σε ποιο man section θα απευθυνθούμε και ποια είναι η πλήρη σύνταξη; _____ **(A12)**

Η κλήση συστήματος `stat()` πόσες παραμέτρους δέχεται, και τι τιμή επιστρέφει; _____ **(A13)**

(C8) Να δημιουργήσετε πρόγραμμα σε C, το οποίο θα δέχεται μια παράμετρο *(αν δε δοθεί παράμετρος θα σταματάει η εκτέλεση)*, και :

(a) θα εκτυπώνει το μέγεθος σε blocks της καταχώρησης (*δηλαδή θα χρησιμοποιηθεί το πεδίο sb->st_size),*

(b) αν είναι αρχείο *(δηλαδή θα χρησιμοποιηθεί το S_IFREG ή το μάκρο* **S_ISREG(sb->st_mode)** *)*, και

(c) αν ο ιδιοκτήτης έχει άδεια εγγραφής σε αυτό, δηλαδή θα χρησιμοποιηθεί το S_IWUSR.

Η μακροεντολή **S_ISREG(sb->st_mode)** επιστρέφει TRUE αν ισχύει. Επίσης, προκειμένου να ελεγχθεί κάποιο συγκεκριμένο bit *(όπως έχει οριστεί στη σελίδα βοήθειας της man)*, χρησιμοποιείται η παρακάτω δομή στην οποία κάνουμε αρχικά λογικό ΚΑΙ με τη μάσκα και στη συνέχεια ισότητα με τη μάσκα. Αν είναι αληθές, τότε το mode είναι το ίδιο με τη μάσκα.
π.χ. If ((sb->st_mode & S_IFSOCK) == S_IFSOCK) { printf(“Object is a Socket”);}.

10. Βελτιστοποίηση Προγράμματος μέσω ανίχνευσης διαρροής μνήμης

Ένα συχνό πρόβλημα των προγραμματιστών, είναι ότι δεν απελευθερώνουν τους πόρους που έχουν δεσμεύσει όταν δε τους χρειάζονται. Για παράδειγμα, κάποιος που δεσμεύει μνήμη με το malloc() ξεχνάει να την απελευθερώσει με το free() όταν δε τη χρειάζεται ή κάποιος που ανοίγει έναν κατάλογο *(opendir())* ξεχνάει να τον κλήσει *(closedir())* στο τέλος.

Ένα από τα προγράμματα που βοηθάνε τον προγραμματιστή να βρίσκει τα memory leaks, είναι το valgrind *(είναι εγκατεστημένο στο zafora).*

Δημιουργήστε ένα νέο αρχείο C με τον παρακάτω κώδικα:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
main(int argc,char *argv[])
{
char *sb,*sb2;
//free(sb);
sb=malloc(100);
sb2=malloc(3);
strcpy(sb,argv[1]);
printf("%s",sb);
free(sb2);
}
```

Κάντε το compile και εκτελέστε το με μια παράμετρο (π.χ. `./a.out testparam`). Φαίνεται ότι λειτουργεί σωστά.

Εκτελέστε στο zafora το πρόγραμμα **`valgrind`** με πρώτη παράμετρο το εκτελέσιμο αρχείο και δεύτερη παράμετρο το string που είχατε βάλει, και θα σας εμφανιστεί μια οθόνη για τη μνήμη που έχει χαθεί.

Πόση μνήμη αναφέρει το **`valgrind`** ότι έχει χαθεί; Κάντε copy paste τις γραμμές που εμφανίζουν αυτή την πληροφορία. _____ **(A14)**

Χρησιμοποιήστε την παράμετρο **`--leak-check=full`** και εκτελέστε πάλι το **`valgrind`** για να δείτε που έχουν χαθεί αυτά τα Bytes.

Σε ποια ακριβώς εντολή έχουν χαθεί τα bytes; Κάντε copy paste τις γραμμές που εμφανίζουν αυτή την πληροφορία. _____ **(A15)**

Συνήθως, το **`valgrind`** εκτελείται με όλες τις σχετικές παραμέτρους για να μας εμφανίσει όσο περισσότερη πληροφορία γίνεται. Αυτές είναι:

`valgrind --tool=memcheck --leak-check=yes`

`--show-reachable=yes --num-callers=20 --track-fds=yes`

Κάντε uncomment τη γραμμή free(sb); μετά compile, και στη συνέχεια εκτελέστε το valgrind. Θα σας εμφανίσει μια ακόμη πληροφορία, ότι το free είναι invalid *(δοκιμάστε το)*. Το **`valgrind`** μπορεί να βοηθάει το προγραμματιστή, στην εύρεση λαθών, ιδιαίτερα σε μεγάλα κομμάτια κώδικα.

Τοποθετήστε ξανά το comment ή διαγράψτε την προβληματική γραμμή.

Το **`valgrind`**, επίσης περιέχει εργαλεία για τη βελτιστοποίηση του συστήματος1

Ένα από αυτά είναι και το cachegrind, που εμφανίζει το ποσοστό αστοχιών στην κρυφή μνήμη του συστήματος.

Εκτελέστε το **`valgrind`** με την παράμετρο **`–tool=cachegrind`** και δώστε το συνολικό ποσοστό ευστοχίας στη μνήμη για την τελευταίου επιπέδου κρυφή μνήμη δεδομένων ***(Last Level Data Cache - LLd)*** και εντολών *(LLi)*. Προσέξτε ότι το πρόγραμμα αναφέρει το miss rate, οπότε θα πρέπει να υπολογίσετε το hit rate. **(A16)**