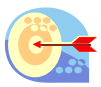
2. Noțiuni pregătitoare – sistemul de operare Linux

2.1. Cuprins modul

Introducere

Scopul principal al acestui laborator este prezentarea sumara a sistemului de operare Unix/Linux. Vor fi discutate si exersate principalele comenzi de operare. Vom prezenta modul de scriere şi compilare a programelor C/C++, dar şi posibilitățile de depanare a programelor. Sistemul de operare Linux este folosit in cadrul laboratorului de pentru a înlesni scrierea programelor prin accesarea unor resurse şi unelte specifice (de exemplu posibilitatea deschiderii mai multor sesiuni la acelaşi terminal). În plus, vine cu elemente noi ce fixează noțiuni deja asimilate din experiența anterioara cu alte sisteme de operare.



Obiective

După parcurgerea acestui laborator studenții vor fi capabili:

- Să utilizeze o serie de comenzi Linux

- Să consulte paginile de manual al comenzilor Linux
- Să editeze, compileze şi ruleze din linia de comandă un program C
- Să depaneze un program folosind gdb/insight

Durată medie de studiu individual

Durata medie de studiu individual : 2 ore

2.1. Prezentarea sistemului de operare UNIX (Linux)

Prima versiune experimentala a sistemului a fost scrisa in 1969 in limbaj de asamblare. Versiunea se numea UNICS (UNiplexed Information and Computing Service). Din cauza pronunției acronimului de mai sus, numele s-a schimbat in UNIX. In 1972 a fost realizat primul compilator pentru limbajul C iar sistemul UNIX a fost rescris in acest limbaj.

Dintre caracteristicile principale ale acestui sistem de operare amintim:

- UNIX este scris in limbajul C => este portabil, sistemul funcționând aproape identic pe mainframe, mini sau microcalculatoare.
- este un sistem deschis: permite folosirea celor mai diverse arhitecturi de calcul.
- este multisesiune: se pot deschide mai multe sesiuni de lucru pe acelaşi terminal.
- este multiproces: se pot rula concurent mai multe procese ce pot comunica intre ele sau pot crea la rândul lor alte procese.
- interpretorul de comenzi (shell) permite introducerea de noi comenzi, combinarea comenzilor, etc. El nu este înglobat in nucleul sistemului de operare si nu este unic.
- permite utilizarea in comun a informațiilor, facilitează lucrul in echipa si comunicația intre utilizatori prin mecanisme evoluate.
- exista pachete de programe pentru orice tip de aplicație.

Sistemul de operare Unix/Linux se compune din 3 parți: nucelul (eng. Kernel), interpretorul de comenzi (eng. shell) şi programele. *Nucleul* reprezintă elementul central al sistemului de operare; el este cel care se ocupa de alocarea de memorie si timp de procesor pentru programele care se executa, gestionează sistemele de fişiere, etc. *Interpretorul de comenzi* reprezintă interfața

dintre utilizator si kernel. *Programele* sunt aplicațiile create de utilizator si executate prin intermediul interpretorului de comenzi.

Pentru dezvoltarea de aplicații este necesara definirea unor anumite standarde. *POSIX* (Portable Operating System for Computer Environment) defineşte interfața intre sistemul de operare si aplicații. Standardul este dezvoltat de IEEE si promovează portabilitatea codului sursa nefiind obligatoriu ca sistemul de operare suport sa fie un sistem UNIX.

Se recomanda ca în realizarea aplicaţiilor din cadrul laboratorului, cat si a temelor de casa, sa folosiţi funcţii din limbajul C care sunt POSIX.

2.2. Interpretorul de comenzi

2.2.1. Gestiunea utilizatorilor

UNIX este un sistem multiutilizator. Un utilizator poate avea mai multe sesiuni deschide pe acelaşi calculator sau pe calculatoare diferite. In sistem exista următoarea ierarhie:

- un super-utilizator - *root* - care este un utilizator privilegiat (admnistratorul de sistem)
- utilizatori normali care au drepturi limitate şi acordate de root. Utilizatorii intra in sistem cu comanda 'login'. Au asociat un nume, o parola, un număr de utilizator.

Exista si grupuri de utilizatori. Informațiile despre utilizatori se găsesc in fisierul /etc/passwd pe care doar super-utilizatorul il poate modifica. Adăugarea/ştergerea de utilizatori în/din sistem o poate face doar super-utilizatorul fie cu comenzile:

```
# useradd
```

```
# rmuser
```

fie prin intermediul unor programe cu interfața grafică în funcție de distribuție.

Comenzile precedate de caracterul # (diez) indică faptul că ele pot fi executate doar de super-utilizator; caracterul # nu face parte din comandă, deci nu-l tastați atunci când introduceți comenzi. Comenzile precedate de caracterul $ (dolar) indică faptul că ele pot fi executate de către utilizatori normali. Pentru a asigura securitatea sistemul folosiți contul de *root* doar atunci când este strict necesar.

Alte comenzi legate de gestiunea utilizatorilor:

$ cat /etc/passwd	permite vizualizarea fişierului cu id utilizatorilor
$ logname	afişează numele utilizatorului curent (numele de login).
$ id	afişează numărul si numele de utilizator si de grup al utilizatorului curent
$ who	afişează numele tuturor utilizatorilor activi la un moment dat (+ informații despre terminalul pe care lucrează si momentul deschiderii sesiunii curente)
$ who am i	afişează datele de mai sus (+numele gazdei) numai pentru utilizatorul curent
$ finger	la fel ca *who* dar cu informații suplimentare (nume complet, adresa, telefon)

2.3. Fişiere

Sistemul de fişiere are o organizare ierarhica, fişierele fiind grupate in cataloage ce alcătuiesc o structura arborescenta. Arborele de fişiere este unic, rădăcina sa fiind simbolizata prin /. Pentru specificarea căii în UNIX se foloseşte / (in MSDOS, Windows se foloseşte \).

De exemplu, avem câteva cataloage dedicate pentru anumite operațiuni:

`tmp`	catalog de fişiere temporare
`bin`	catalog de fişiere executabile (pentru diferite comenzi)
`dev`	catalog pentru fişierele asociate perifericelor
`home`	catalogul cu fişierele utilizatorilor

Pentru a lista fişierele din catalogul curent executați comanda `ls`.

2.3.1. Clasificarea fişierelor

1. fişiere obişnuite: şiruri de octeţi terminate cu `^D`

2. fişiere de tip catalog: tabela cu o intrare pentru fiecare fişier din catalog. Fiecare catalog are doua intrări suplimentare:

- „.” referință către respectivul catalog (un singur punct, fără ghilimele)
- „..”referință spre catalogul părinte (două puncte, fără ghilimele)

3. fişiere speciale: asociate dispozitivelor de I/O (in catalogul /dev)

`/dev/fd0`	prima unitate de disc flexibil existenta
`/dev/lp0`	prima imprimanta sau primul port paralel
`/dev/tty0`	primul din cele 12 terminale virtuale
`/dev/ttyS0`	prima interfața seriala (COM1 în MS-DOS, Windows)

Fişierele speciale pot fi de tip bloc sau caracter in funcție de tipul de terminal asociat. In UNIX, unui dispozitiv de I/O i se asociază doua numere:

- major - indica tipul de dispozitiv (ex.: imprimanta).

- minor - al câtelea dispozitiv periferic este din clasa de dispozitive de acelaşi tip (ex.: a doua imprimantă).

4. fişierele FIFO ("named pipe"): sunt fişiere speciale utilizate in comunicaţia dintre procese prin mecanismul pipe. Diferă de celelalte doar prin modul de acces la informaţie: accesul se poate face o singura data (fără posibilitatea de revenire) iar politica ce gestionează acest acces este FIFO (First In First Out).

5. socket-i: sunt fişiere folosit in comunicaţia dintre procese prin reţea. Un socket poate fi utilizat si pentru comunicaţia intre procesele de pe acelaşi calculator gazda.

6. legătura simbolică: fişier ce conţine un pointer spre un alt fişier.

2.3.2. Permisiuni (drepturi) de acces

Drepturile de acces sunt asociate fiecărui fişier in parte si sunt in număr de trei:

r - drept de citire in acel fişier

w - drept de scriere in acel fişier

x - drept de a executa acel fişier

Utilizatorii se clasifica in trei categorii, in funcție de relația față de un fişier:

- user (proprietarul fişierului - este unic)
- group (utilizatorii din acelaşi grup cu proprietarul fişierului)
- others (ceilalți utilizatori)

Deci vor trebui sa existe 9 poziții pentru precizarea completa a drepturilor. Pentru fişierele de tip catalog aceste drepturi au alte semnificații:

r - drept de listare a conținutului catalogului;

w - se pot crea/şterge fişiere in/din catalog;

x - se poate parcurge catalogul pentru căutarea unui fişier si exista drept de acces la acel fişier.

Pentru a lista fişierele din catalogul curent împreună cu drepturile lor de acces executați comanda *ls –all .*

2.3.3. Modificarea permisiunilor de acces

```
# chmod [pentru_cine] tip_modificare nume_fisier
```

pentru_cine poate fi:

u = schimb drepturile pentru utilizator (user)

g = schimb drepturile pentru grup (group)

o = schimb drepturile pentru ceilalți (others)

a = schimb drepturile pentru toți (all)

tip_modificare:

are minim doua caractere:

primul caracter: + se adăuga nişte drepturi (plus)

- se retrag niste drepturi (minus)

= se seteaza drepturile (egal)

al doilea caracter: r, w sau x

Exemple

`# chmod a=x fisier`	Se setează drept de execuție pentru toata lumea asupra fişierului fis
`# chmod go+wx fisier`	Se atribuie utilizatorilor din acelaşi grup cu proprietarul si celorlalți utilizatori drept de scriere si execuție a fişierului `fisier`
`# chmod 754 fisier`	Se setează asupra fişierului *fis* drepturile: 7 = r+w+x pentru *user* 5 = r+x pentru *group* 4 = r pentru *others*
`# chown proprietar_nou fisier`	Schimbă proprietarul unui fişier
`# chgrp grup_nou fisier`	Schimbă grupul unui fişier

2.4. Comenzi de shell

- Un prim exemplu de comanda *shell* pentru lucrul cu fişiere o reprezintă `ls`. Aceasta fără nici o opțiune permite listarea conținutului directorului in care va aflați curent. Exista însă fişiere ascunse (cele al căror nume începe cu ".") care se pot afişa doar prin

  ```
  $ ls -a
  ```

  În acest caz `ls` este exemplu de comandă, pe când `-a` este exemplu de opțiune care funcționează pentru o anumita comanda. Aproape toate comenzile shell accepta opțiuni.

- Alta comanda de shell este `mkdir` (make directory). Comanda permite crearea de noi directoare.
- O alta comanda utila o reprezintă `cd` (change directory). Ca efect al acestei comenzi directorul curent (directorul in care va aflați la un anumit moment) se poate schimba. Exista si doua directoare mai speciale, si anume "**.**" - care reprezintă directorul in care va aflați

curent si "**..**" - care reprezintă directorul aflat cu un nivel mai sus in ierarhia de directoare. De exemplu daca încercați "*cd .*" veți constata ca rămâneți in cadrul aceluiaşi director in care erați si înainte de emiterea comenzii. Pentru a afla in care director din cadrul ierarhiei va aflați la momentul de timp curent puteți folosi comanda `pwd`. Ca rezultat al execuției acestei comenzi vi se afişa pe ecran calea completa pana la directorul in care va aflați. Ca exercițiu puteți folosi comenzile de pana acum pentru a explora sistemul de fişiere.

- O alta comanda o reprezintă comanda `cp` (copy)**.** Aceasta va permite copierea unuia sau mai multor fişiere.
- Comanda `mv` vă permite mutarea locului unuia sau a mai multor fişiere in cadrul sistemului de fişiere.
- Pentru a crea un fişier folosiți comanda `touch nume_fisier` sau mai simplu `>nume_fis`
- Pentru ştergerea unui fişier puteți folosi una dintre comenzile `rm` sau `rmdir`.
- Pentru vizualizarea conținutului unui fişier se poate folosi comanda `cat`. Ca rezultat al acestei comenzi conținutul fişierului primit ca argument se va afişa în întregime pe ecran daca nu se folosesc alte opțiuni.
- Comanda `less` va scrie de asemenea conținutul fişierului pe ecran, însă pagina cu pagina. Folosiți tasta *spaţiu* pentru a trece de la pagina la pagina si tasta *q* pentru a determina terminarea afişării.
- Comanda `head` fără argumente va afişa primele 10 linii din fişier pe ecran.
- Comanda `tail` fără argumente va afişa ultimele 10 linii din fişier pe ecran.
- Pentru a căuta un anumit cuvânt in cadrul fişierului se poate folosi `less`, după care folosiți secvența "*/[nume_de_cautat]*" pentru a va poziționa pe prima apariție in text a cuvântului respectiv, dacă exista. Apoi pentru a trece la următoarea apariție a cuvântului in text folosiți tasta *n.*
- Un utilitar destul de folositor este şi `wc`, care va permite diverse contorizări. De exemplu pentru a afla numărul total de cuvinte dintr-un fişier folosiți `wc -w fisier`, sau pentru a afla numărul de linii din cadrul unui fişier folosiţi comanda `wc -l fisier`.
- Toate aceste comenzi si multe altele incluse in sistemul de operare Unix/Linux au disponibilă şi o documentație, care se poate vizualiza folosind comanda `man`. De exemplu pentru a afla care sunt opțiunile comenzii *wc* folosiți `man wc`. Ca rezultat pe ecran vi se va afişa lista de parametrii ai comenzii cu explicații legate de folosirea acestora.
- Ca exercițiu de laborator folosiți comanda *man* pentru a studia comenzile de pana acum mai in profunzime.

2.5. Compilatorul GCC

In cadrul laboratorului veți primi o serie de teme in limbajul C/C++. Sistemul de operare Unix/Linux dispune de un utilitar pentru compilarea fişierelor scrise in aceste limbaje de programare, numit **gcc**. Ca alternative la *gcc* mai exista si *g++, cc* si *CC.* In continuare se va descrie modul in care se foloseşte utilitarul *gcc.*

Sa presupunem ca avem un fişier sursa numit hello.c. Comanda pentru compilarea standard a acestui fişier este *gcc hello.c.* Ca rezultat se obține un fişier **a.out** care se poate executa executând *./a.out*

Daca insa compilam folosind *gcc* hello.c -o hello se va obține fişierul executabil **hello**.

Puteți insa folosi si o alta secvența de compilare, si anume daca introduceți

```
gcc -c hello.c
gcc hello.o -o hello
```

Rezultatul obținut va fi acelaşi ce cel obținut prin `gcc hello.c -o hello`. Ceea ce diferă este faptul ca prima comandă din cadrul secvenței va *compila* fisierul *hello.c* intr-un fişier cod maşina numit *hello.o*, iar cea de-a doua comandă va *lega* hello.o cu unele dintre bibliotecile sistemului pentru a produce rezultatul final, si anume fişierul executabil *hello* care poate fi executat folosind ./hello

De asemenea compilatorul va permite compilarea simultana a mai multor fişiere sursa pentru obținerea ulterioara a unui fişier executabil. De exemplu

```
gcc fisier1.c fisier2.c -o program
```

va compila cele doua fişiere sursa si va furniza un fişier executabil numit *program*.

Compilatorul *gcc* suporta o serie de opțiuni de compilare (pentru studierea acestora mai in amănunt se va folosi **man gcc**). Optiune **-g** va permite obținerea unui executabil care va conține informații de *depanare (debug)*. Aceasta opțiune este utila in combinație cu un program de depanare cum ar fi **gdb**, care de asemenea se va studia in cadrul laboratorului. O alta opțiune utila este **-Wall**. Aceasta opțiune specifică compilatorului sa afişeze in timpul compilării toate avertismentele (*Warnings*) despre secvențe de cod corecte din punct de vedere sintactic, dar care ar putea conține totuşi erori logice. O alta opțiune utila este **-l**. Aceasta opțiune permite legarea de biblioteci ale sistemului de operare. De exemplu, daca programul conține funcții matematice cum ar fi *sqrt* atunci programul se compilează folosind `gcc program.c -o program -lm`.

2.6. Depanarea programelor

Un program de depanare (*debugging*) este un program folosit pentru rularea altor programe astfel încât utilizatorul sa poată exercita control asupra execuției programului. Utilizatorul poate de exemplu examina variabilele din program la apariția unor erori. In Linux program de depanare este **gdb**. Pentru folosirea acestuia programul trebuie sa fi fost compilat cu opțiunea **-g** (după cum am spus mai sus). După compilare se porneşte programul gdb având ca argument numele programului de executat

```
$ gdb program_executabil
```

Ca efect *gdb* va introduce intr-o interfața text de unde puteți alege o serie de opțiuni. In continuare sunt descrise o serie de comenzi pe care le puteți tasta in cadrul acestei interfețe.

O prima comanda este **run**. Acesta lansează in execuție programul. Comanda primeşte si argumente, pe care le transfera programului care rulează.

O alta comanda este **break**. Aceasta primeşte ca argument locul in care se creează un punct de oprire in cadrul execuției. Programul se va opri automat atunci când ajunge la un astfel de punct de control.

De exemplu unele dintre cele mai frecvente locuri de oprire sunt cele de la începutul funcțiilor. De exemplu *break Functie* va forma un punct de control la începutul funcției având numele *Functie.* Puteți folosi ca argument si un număr, ceea ce înseamnă linia la care se va forma punctul de control. In conjuncție cu aceasta comanda se poate folosi comanda **delete**, care are ca efect ştergerea unui punct de control generat anterior.

Comanda **step** are ca efect execuția următoarei instrucțiuni din cadrul programului. După execuția acesteia programul se va opri la următoarea instrucțiune de executat.

Similar comanda **next** are acelaşi efect, însă ca diferență daca următoarea instrucțiune este reprezentată de apelul unei funcții aceasta se tratează unitar ca o singura instrucțiune.

Comanda **finish** are ca efect execuția unor comenzi *step* repetate pana când se ajunge la sfârşitul funcției curente.

Comanda **continue** are ca efect continuarea execuției programului pana la întâlnirea unui punct de control sau pana la terminarea programului.

Comanda **where** afişează şirul de apeluri de funcții prin care s-a ajuns la instrucțiunea curenta.

Comanda **print** primeşte ca argument o expresie C (de obicei numele unei variabile este suficient) si va afişa valoarea acestei expresii in cadrul curent al programului.

Atunci când nu sunteți siguri de formatul unei comenzi *gdb* mai exista si comanda **help** care afişează informații despre comenzi gdb.

Pentru terminarea execuției programului gdb puteți folosi comanda **quit.** Utilizarul de depanare *gdb* dispune şi de o interfață grafică, denumită *insight*. Aceasta trebuie instalată separat.

2.7. Fişiere Makefile

Un utilitar util in Linux este **make**. Acesta citeşte instrucțiuni din cadrul unui fişier text pe care apoi le executa. Implicit *make* citeşte instrucțiunile din fişierul text cu numele *Makefile*, însă se poate specifica numele oricărui fişier text.

In cadrul fişierului Makefile instrucțiunile sunt scrise pe fiecare linie. In cazul in care instrucțiunile sunt prea lungi se poate folosi caracterul **"\"** urmat de *Enter*, efectul fiind acela ca următoarea linie se considera a fi o continuare a liniei curente. Comentariile in cadrul fişierului *Makefile* încep cu caracterul **"#"**. După acest caracter tot ce urmează pana la sfârşitul liniei se considera comentariu si se ignora, excepție făcând caracterul *"\"*.

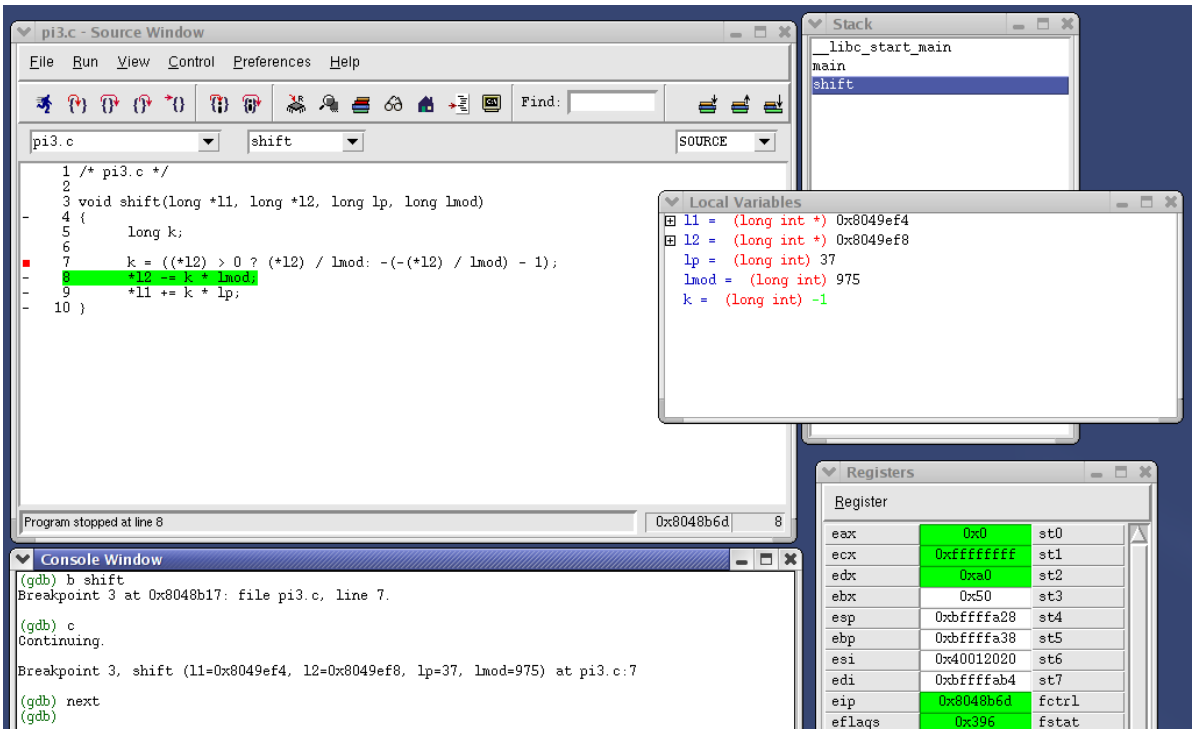


Figura 2.1 Utilitarul de depanare *insight*

De exemplu

```
linia_unu \

linia_doi # comentariu \

linia_trei
```

este similar cu

```
linia_unu linia_doi linia_trei
```

Conceptul de **regula** in cadrul fişierelor *Makefile* specifica cum si când se executa o secventa de instrucțiuni. De exemplu sa presupunem ca avem un proiect care implica compilarea fişierelor sursa *main.c* si *io.c* si producerea fişierului executabil *program*. Un exemplu de fişier Makefile care ne ajuta in cadrul proiectului nostru atunci ar fi:

```
project: main.o io.o
          #comenzile incep cu doi TAB-i
          gcc main.o io.o -o project
main.o: main.c
          gcc -c main.c
io.o: io.c
          gcc -c io.c
```

Exemplul poate părea un pic forțat, dar este folositor pentru a înțelege mai bine conceptul de regula. In exemplu acesta avem de a face cu trei reguli, una pentru generarea fişierului executabil, o alta pentru compilarea fişierului sursa `main.c` si o a treia pentru compilarea fişierului sursa io.c.

Liniile care au in componenta **":"** se numesc linii de dependenta. Ceea ce se afla la stânga de ":" reprezintă numele dependentei (sau a regulii), pe când ceea ce se afla la dreapta reprezintă regulile de care depinde regula curenta. De exemplu linia `project: main.o io.o` specifica faptul ca regula de obținere a fişierului executabil depinde de regulile de formare a fişierelor main.o si io.o. La rulare *make* compara momentul de timp când fişierul *project* a fost ultima oara modificat cu momentul de timp la care fişierele main.o si/sau io.o au fost modificate pentru a determina dacă este necesara o recompilare sau nu a vreunuia dintre fişierele sursa.

Sa presupunem in exemplul nostru mai departe ca atât `main.c`, cat şi `io.c`, depind ambele mai departe de *def.h*. In cazul acesta se pot include dependente adiţionale, ca de exemplu putem adaugă la exemplul nostru linia

```
main.obj io.obj : incl.h
```

In fişierul Makefile pot fi incluse si linii de macro-definiţii. O linie de macro-definiţie este o linie care are un nume de macro-definiţie, semnul **"="** si o valoare a macro-definiţiei. In cadrul fişierului expresiile de forma *${nume}* sau *$(nume)* se înlocuiesc cu valorile asociate acelor nume. Daca numele este compus dintr-un singur caracter atunci parantezele pot fi omise. Ca exemplu:

```
CC    = gcc
FLAGS = -g #pentru depanare
project: main.o io.o
          $(CC) main.o io.o -o project
main.o: main.c
          ${CC} -c main.c
io.o: io.c
          gcc -c io.c
```

Orice sistem de operare pune la dispoziția programatorilor o serie de *servicii* prin intermediul cărora acestora li se oferă acces la resursele hardware si software gestionate de sistem: lucrul cu tastatura, cu discurile, cu dispozitivul de afişare, gestionarea fişierelor si directoarelor etc. Aceste servicii se numesc *apeluri sistem*. De cele mai multe ori, operațiile pe care ele le pot face asupra resurselor gestionate sunt operații simple, cu destul de puține facilitați. De aceea, frecvent, se pot întâlni in bibliotecile specifice limbajelor de programare colecții de funcții mai complicate care gestionează resursele respective, dar oferind programatorului niveluri suplimentare de abstractizare a operațiilor efectuate, precum si importante facilitați in plus. Acestea sunt *funcțiile de biblioteca*. Trebuie subliniat faptul ca funcțiile de biblioteca cu ajutorul cărora se poate gestiona o anumita resursa sunt implementate folosind chiar funcțiile sistem corespunzătoare, specifice sistemului de operare.

Bibliografie

Referinţe web (unele sunt vechi, s-ar putea sa nu mai functioneze)

Site-ul oficial Linux - http://www.linux.org/

Site-ul oficial Unix - http://www.unix.org/

Un site dedicat utilizatorul Linux, foarte util si pentru începători - http://www.linux.ro/

O istorie a Linux-ului - http://www.li.org/linuxhistory.php; http://www.xplinux.biz/support/history.htm

Shell pentru incepatori - http://www.freeos.com/guides/lsst/The Linux Terminal - a Beginners' Bash - http://linux.org.mt/article/terminal

An A-Z Index of the Linux BASH command line - http://www.ss64.com/bash/Linux MAN Pages Indexed HTML Version - http://linux.ctyme.com/

LINUX MAN PAGES ONLINE - http://man.he.net/

Welcome to the GCC home page - http://gcc.gnu.org/

The Linux GCC HOWTO - http://www.faqs.org/docs/Linux-HOWTO/GCC-HOWTO.html

MinGW - http://www.mingw.org/

Debugging with GDB - http://www.delorie.com/gnu/docs/gdb/

GDB: The GNU Project Debugger - http://www.gnu.org/software/gdb/documentation/

GDB Tutorial - http://www.cs.princeton.edu/~benjasik/gdb/gdbtut.html

Insight tutorial - http://blog.copyninja.info/2008/10/gdbs-gui-insight-tutorial.html

Make - a tutorial - http://www.eng.hawaii.edu/Tutor/Make/

How to write a Makefile - http://vertigo.hsrl.rutgers.edu/ug/make_help.html

The Makefile - http://www.opussoftware.com/tutorial/TutMakefile.htm

Kate (KDE Advanced Text Editor) - http://kate.kde.org/