Laborator 1 - Sisteme de Operare

Ce este un sistem de operare ?

Un sistem de operare este un produs software ce se ocupa de gestionarea si coordonarea activitatilor unui sistem de calcul. El mediaza accesul programelor de aplicatie la resursele masinii.

Moduri de interactiune cu sistemul de operare

Un utilizator poate interactiona cu sistemul prin doua mijloace:

1. Mediul grafic (in engleza *Graphical User Interface* sau *GUI*)
2. Mediul linie de comanda (*Command Line Interface*)

Pe tot parcursul laboratoarelor vom lucra foarte mult in interfata linie de comanda deoarece este rapid de folosit si arata la fel pe orice distributie Linux.

Important !!! Interfata in linie de comanda permite realizarea tuturor operatiilor ce pot fi realizate si din interfata grafica.

Shell-ul reprezinta un produs software ce asigura interfata cu utilizatorul. Shell-ul este rulat de catre o consola. Aceasta consola poarta numele de "***terminal"*** pe distributile Linux. Exista mai multe tipuri de shell in Linux, cel mai folosit tip fiind **bash-ul** (*Bourne Again Shell*).

Shell-ul sau **interpretorul de comenzi** ofera utilizatorului un prompt. Acesta are urmatorul format:

username@localhost:~$

unde:

- ***username*** reprezinta numele utilizatorului logat la acel terminal
- ***localhost*** reprezinta numele statiei de lucru
- ***"~"*** indica directorul curent corespunzator home-ului utilizatorului. Deobicei este /home/username
- ***"$"*** marcheaza terminarea promptului si inceperea zonei unde utilizatorul poate introduce comenzi

Exemplu: andrei@andrei-PC:~$

Comenzile sunt cuvinte cheie pe care utilizatorul le introduce in interpretorul de comenzi pentru a configura sistemul sau a obtine anumite rezultate.

O comanda poate sa fie simpla (adica formata dintr-un singur cuvant) sau compusa (contine argumente suplimentare separate prin spatiu si care sunt marcate de multe ori prin "-").

Exemplu:

$ ls (listeaza continutul directorului curent)

$ ls -l (listeaza informatii suplimentare despre intrarile din director cum ar fi dimensiunea, data ultimei modificari, etc.)

Pentru a afla informatii detaliate despre o comanda se poate consulta pagina de manual Linux prin:

$ man nume_comanda

sau

$ nume_comanda --help

Executia unei comenzi inseamna, de obicei, generarea unui proces dintr-un executabil asociat comenzii.

Pentru a parcurge comenzile din istoric se foloseste tasta <Sageata-sus>

Pentru a folosi facilitatile de auto-complete a bash-ului se foloseste tasta <Tab>. Dupa introducerea unei bucati a numelui comenzii restul se poate completa automat prin apasarea tastei <Tab>.

Comenzi pe care le vom folosi frecvent:

$ ls

$ cd cale/nume_director (schimba directorul curent cu cel specificat ca argument)

$ cat nume_fisier (afiseaza continutul fisierului)

$ echo mesaj (afiseaza mesajul la standard-output)

$ grep sir fisier (cauta sirul “sir” in fisierul “fisier” si afiseaza linile pe care se afla)

$ rm fisier (sterge fisierul cu numele “fisier”)

$ mkdir director (creeaza un director cu numele “director”)

Alte comenzi le vom invata pe masura ce avansam cu materia. Pentru moment, a fost important sa intelegem notiunile fundamentele de comunicare cu sistemul prin interfata in mod linie de comanda.

Limbajul de programare C

In cadrul laboratorului de sisteme de operare, limbajul de programare folosit va fi C. Acesta este menit sa simplifice scrierea programelor apropiate de masina. A fost dezvoltat de catre Dennis Ritchie intre 1969-1973 la Bell Labs in Statele Unite si folosit pentru rescrierea in intregime a sistemului de operare Unix. C este in continuare foarte folosit la scrierea sistemelor de operare. Nucleele sistemelor Windows si Linux sunt scrise in C.

GCC

GCC-ul (GNU Compiler Collection) este o suita de compilatoare disponibila pe majoritatea distributilor Linux. Este folosit pentru a compila o paleta larga de limbaje de programare precum C, C++, Java, Fortran, Objective-C.

Un **fisier sursa** reprezinta un fisier text in care este scris cod corespunzator unui anumit limbaj de programare. Acest fisier este scris/editat folosind un editor de text (Notepad, Sublime Text, Gedit etc).

Dintr-un fisier sursa C se obtine prin procesul de compilare un fisier executabil. Compilarea se realizeaza in Linux prin comanda **gcc** pentru fisiere sursa C si **g++** pentru fisiere sursa C++.

```
$ gcc  main.c
```

sau

```
$ g++ main.cpp
```

In urma acestor comenzi va aparea un fisier denumit implicit **a.out.** Acesta este un fisier executabil ce poate sa fie lansat prin comanda:

```
$ ./a.out
```

Daca se doreste obtinerea unui executabil cu alt nume se poate folosi optiunea **-o**

```
$ gcc main.c -o progamul-meu
```

sau

```
$ g++ main.cpp -o programul-meu
```

Fazele compilarii

Intern, gcc-ul trece prin mai multe faze de prelucrare a fisierului sursa pana la obtinerea executabilului.

Folosind diverse optiuni, putem opri compilarea la una din fazele intermediare.

1. Precompilarea sau Preprocesarea

In urma procesului de preprocesare, se realizeaza substitutii in fisierul sursa la intalnirea comenzilor de precompilare ce incep cu caracterul “#” in prima coloana (#define,#include, etc...). Pentru a opri gcc-ul la aceasta faza introducem comanda:

```
$ gcc -E main.c
```

2. Compilarea

Compilarea este faza in care din fisierul preprocesat se obtine un fisier în limbaj de asamblare.

Pentru a opri gcc-ul la aceasta faza introducem comanda:

$ gcc -S main.c

3. Asamblarea

Asamblarea este faza in care codul scris in limbaj de asamblare este tradus in ***cod* masina** reprezentand codificarea binara a instructiunilor programului initial. Fisierul obtinut poarta numele de fisier **cod obiect**. Pentru a opri gcc-ul la aceasta faza introducem comanda:

$ gcc -c main.c

Din motive ce nu o sa fie explicate aici, este importanta folosirea optiunii **-fPIC** atunci cand oprim compilarea la faza de asamblare. Astfel, comanda arata in felul urmator:

$ gcc -fPIC -c main.c

4. Editarea de legaturi

Pentru obtinerea unui fisier executabil este necesara rezolvarea diverselor simboluri prezente în fisierul obiect. Aceasta operatie poarta denumirea de **editare de legaturi**, **link-editare**, **linking** sau **legare**.

Un simbol poate sa fie un nume de functie sau de variabila a carui definitie nu e stiuta. Pentru a afla simbolurile pe care un fisier obiect nu le are rezolvate folosim comanda **nm** astfel:

$ nm main.o

Simbolurile marcate cu “T” sunt simboluri rezolvate, iar cele marcate cu “U” sunt simboluri nerezolvate.

Sa presupunem ca in fisierul main.c sunt declarate si folosite 2 functii , f1 si f2. Acestea au definitia scrisa in doua fisiere sursa separate, f1.c si f2.c. Daca dam comanda:

$ gcc main.c

Compilarea va ceda la faza de linkare deoarece nu cunoaste definitia celor 2 simboluri (f1 si f2). Comanda corecta este:

$ gcc main.o f1.o f2.o

f1.o si f2.o sunt fisierele obiect obtinute din preprocesarea, compilarea si asamblarea fisierelor sursa in format text f1.c si f2.c. Comanda se poate da si direct pe fisierele sursa astfel:

$ gcc main.c f1.c f2.c

Dar am folosit exemplul cu fisiere obiect in scopuri didactice.

ATENTIE !!!

int f(int a); // Declararea functiei

```
int f(int a)
{
   std::cout<<”a= ”<<a;  // Definirea functiei
}
```

Linker-ul poate sa fie invocat de catre gcc (cum se intampla cand am dat comanda de mai sus) sau chemat explicit prin urmatoarea comanda:

$ ld main.o f1.o f2.o

ATENTIE !!!

Comanda **ld** nu duce fisiere sursa la fisiere obiect ci doar realizeaza legarea mai multor module obiect si creeaza executabilul.

Biblioteci in Linux

O biblioteca este o colectie de fisiere obiect. In momentul in care un program are nevoie de o functie, linker-ul va apela respectiva functie din biblioteca. Numele fisierului reprezentand biblioteca trebuie să aibă prefixul **lib**.

In Linux bibliotecile sunt de 2 feluri:

- statice, au extensia **.a**
- partajate, au extensia **.so**

Diferenta majora intre ele este aceea ca executabilul realizat in urma legari cu biblioteci statice contine o copie a codului functiei fapt ce face ca executabilul sa creasca in dimensiune pe cand functile din bibliotecile partajate sunt invocate dinamic la momentul executiei de catre un program numit **loader.**

Considerand ca avem 3 functii (f1, f2 si f3), definite in 3 fisiere sursa (f1.c, f2.c si f3.c), din care obtinem 3 fisiere obiect (f1.o, f2.o si f3.o). Dorim sa organizam aceste module intr-o biblioteca. Comenzile corespunzatoare pentru crearea bibliotecilor sunt:

$ ar rc libintro_static.a f1.o f2.o f3.o

pentru **biblioteci statice** si:

$ gcc -shared f1.o f2.o. f3.o -o libintro_shared.so

pentru **biblioteci partajate**.

Majoritatea bibliotecilor puse la dispozitie de catre compilator sunt biblioteci partajate.

ATENTIE!!!

La comanda:

$ gcc main.c

Se face linkarea automat cu biblioteca **libc.so** ce se afla in /usr/lib/x86_64-linux-gnu/libc.so

Alte optiuni utile pentru gcc:

- `-L`*`cale`* – instruieste compilatorul sa caute si in directorul `cale` bibliotecile pe care le foloseste programul; optiunea se poate specifica de mai multe ori, pentru a adauga mai multe directoare
- `-l`*`biblioteca`* – instruieste compilatorul ca programul are nevoie de biblioteca *biblioteca*. Fisierul ce contine biblioteca trebuie sa se numeasca *libbiblioteca.so* sau *libbiblioteca.a*.
- `-I`*`cale`* – instruieste compilatorul sa caute fisierele antet (headere) si in directorul `cale`; opțiunea se poate specifica de mai multe ori, pentru a adauga mai multe directoare.

Link-uri utile:

http://www.tutorialspoint.com/cprogramming/index.htm

pentru recapitulare limbajul C.

https://books.google.ro/books?id=_JFGzyRxQGcC&printsec=frontcover&hl=ro&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

pentru a aprofunda notiunile de la laborator.