Lab 1

Comenzi linux de baza

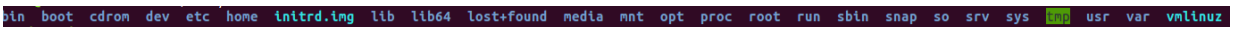
1. **Listati continutul directoarelor /, /bin, /usr, /etc, /usr/include. Acolo unde este cazul, paginati listarea (ls | less). cautati, in fisierul /usr/include/stdio.h, textul *printf*.**

   Pentru acest task vom folosi programul ls(list). Acest program listeaza continul directorului dat ca si parametru, sau daca este rulat fara parametri, listeaza continutul directorului curent.

   De exemplu, in cazul in care rulam programul din folderul home (~):

   ```
   ls /
   ```

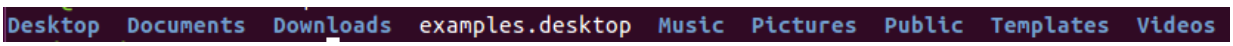
   Va printa:

   Daca rulam programul fara parametri:

   ```
   ls
   ```

   Acesa va afisa:

   In Linux, **/** e scurtatura pentru folderul root, si **~** e scurtatura pentru folderul home al userului curent.*

Schimbarea directorului curent se face prin comanda **cd** (change directory). Exemplu de uz:

```
#change current directory to user home
cd ~
#change current directory to root/bin folder
cd /bin
#cd can be used to navigate a whole directory structure
cd ~/this/works/as/long/as/the/folders/actually/exist
#cd can be used to navigate to an upper folder level
cd ..
```

Paginarea se face prin comanda:

```
ls | less
```

Aceasta redirectioneaza outputul programul **ls** la program **less** prin **|** (pipe). Redirectionarea outputului este un concept fundamental in Linux. Astfel se pot scrie programe specializate pe un singur lucru iar outputul acestor programe se poate redirectiona pentru prelucrare catre alte programe. Prin acest concept putem executa comenzi specializate pentru cerintele noastre, comenzi ce nu ar fi posibile daca am executa un singur program.

Pentru a iesi din programul **less**, se apasa tasta *q*.

Pentru a cauta text intr-un fisier, se va folosi utilitarul **grep**. Primul parametru reprezinta text ul cautat iar al doilea reprezinta fisierul in care sa cautam. Exemplu de uz:

```
grep "text to search" path/to/file
```

Acesta parseaza fisierul dat ca si parametru si printeaza fiecare linie in care textul cautat a fost gasit.

Dupa executarea programului **grep** pe fisierul *"stdio.h"* outputul va fi:

```
extern int fprintf (FILE *__restrict __stream,
extern int printf (const char *__restrict __format, ...);
extern int sprintf (char *__restrict __s,
extern int vfprintf (FILE *__restrict __s, const char *__restrict __format,
extern int vprintf (const char *__restrict __format, _G_va_list __arg);
extern int vsprintf (char *__restrict __s, const char *__restrict __format,
extern int snprintf (char *__restrict __s, size_t __maxlen,
     __THROWNL __attribute__ ((__format__ (__printf__, 3, 4)));
extern int vsnprintf (char *__restrict __s, size_t __maxlen,
     __THROWNL __attribute__ ((__format__ (__printf__, 3, 0)));
extern int vasprintf (char **__restrict __ptr, const char *__restrict __f,
     __THROWNL __attribute__ ((__format__ (__printf__, 2, 0))) __wur;
extern int __asprintf (char **__restrict __ptr,
     __THROWNL __attribute__ ((__format__ (__printf__, 2, 3))) __wur;
extern int asprintf (char **__restrict __ptr,
     __THROWNL __attribute__ ((__format__ (__printf__, 2, 3))) __wur;
extern int vdprintf (int __fd, const char *__restrict __fmt,
     __attribute__ ((__format__ (__printf__, 2, 0)));
extern int dprintf (int __fd, const char *__restrict __fmt, ...)
     __attribute__ ((__format__ (__printf__, 2, 3)));
extern int obstack_printf (struct obstack *__restrict __obstack,
     __THROWNL __attribute__ ((__format__ (__printf__, 2, 3)));
extern int obstack_vprintf (struct obstack *__restrict __obstack,
     __THROWNL __attribute__ ((__format__ (__printf__, 2, 0)));
```

2. **Creati, in directorul personal (~), urmatoarea structura de directoare si fisiere:**

```
(dir. personal)
|
so
|
lab1
|
|
+-- abc
|    +-- x (fisier)
|    +-- y (fisier)
|    +-- t1 (fisier)
|    +-- t2 (fisier)
|    +-- t3 (fisier)
|    +-- t (director)
|        +-- a (fisier)
|        +-- b (fisier)
|
+-- zz (director)
|    +-- x (fisier)
|
+-- tt (director)
```

Pentru a crea un fisier in linux, se foloseste utilitarul **mkdir**. Exemplu de uz:

```
mkdir folder_to_be_created
```

Pentru a crea o structura de directoare, se foloseste parametru *-p*.

```
mkdir -p /directory/structure/we/need
```

Pentru a crea un fisier, se folosete comanda **touch**.

```
#create a file
touch file_to_be_created
#create a file in another folder
touch ~/directory/file_to_be_created
```

3. **Copiati directorul abc cu tot continutul sau (recursiv) ca subdirector al lui zz (va rezulta un subdirector abc in zz)**

   In Linux, pentru a copia, se foloseste utilitarul **cp**. Pentru a copia un director recursiv, se foloseste parametru *-r*.

```
#copy file
cp ~/source/file ~/destination
#copy folder recursively
cp -r ~/source ~/destination
```

4. **Copiati continutul lui abc in directorul zz fara a suprascrie fisierele cu acelasi nume (x, in cazul nostru)**

   Pentru a vedea atributele unui fisier (cand a fost creat, permisiuni etc.) putem folosi:

```
#list files and attributes
ls -l
```

   Astfel, prin timestampul afisat, vom putea verifica daca comanda noastra de copiere a mers.

   Pentru a copia continutul unui director, si nu directorul in sine putem folosi caracterul variabil `*`:

```
#copy all files inside a folder
cp ~/source/* ~/destination
```

   In final, pentru a copia tot continutul unui director (recursiv), fara a suprascrie fisiere deja existente in directorul destinatie, folosim paramentru *-n* (no overwrite) in combinatie cu parametru deja folosit -r.

```
cp -rn ~/source/* ~/destination
```

5. **Copiati fisierele t1 si t2 din abc in tt**

Permisiuni pentru directoare si fisiere

In Linux fiecare fisier si director are asignat anumite permisiuni pentru a stabili cine poate citi, scrie sau executa. Pentru a afla permisiunile unui fisier sau director, putem folosi comanda **ls -l** .

Pentru un fisier, comanda va afisa:

```
-rw-rw-r-- 1 vlad vlad    0 mar  3 20:35 file
```

Permisunile sunt grupate cate trei astfel:

- **owner** - primele 3 litere, acestea au un impact asupra permisiunilor ce se aplica asupra creatorului fisierului sau folderului. Nu afecteaza nici un alt user.
- **group** - urmatoarele 3 litere, au un impact asupra grupurilor care au fost asignate la fisierul sau directorul in cauza. Nu afecteaza nici un alt user.
- **all users** - ultimele 3 litere, aceste permisiuni se aplica pentru toti utilizatorii unui sistem.

Tipuri de permisiuni:

- **read(r)** - specifica daca un user poate citi fisierul sau directorul
- **write(w)** - specifica daca un user poate modifica fisierul sau directorul
- **execute(x)** - specifica daca un user poate executa fisierul sau directorul

Pentru un director, comanda va afisa:

```
drwxrwxr-x 2 vlad vlad 4096 mar  3 20:35 folder
```

Prima litera, *"d"*, specifica faptul ca acesta este un director. In contextul unui director permisiunea *"x"* specifica daca directorul poate fi accesat. In exemplul de mai sus, atat creatorul, cat si grupurile care au fost asignate la acest director il pot citi, modifica sau accesa, orice alt utilizator este restrictionat doar la citire si accesare.

1. **Creati un director. In acest director creati un fisier. Schimbati permisiunile pentru director astfel incat sa aveti drept de executie dar nu si de citire. Rulati comnda *ls*, ce observati? Schimbati permisiunile astfel incat sa aveti drept de citire, dar nu de executie. Ce observati?**

   In Linux, permisiunile se pot controla cu utilitarul **chmod**. In lista de parametri grupurile sunt prescurtate in **u**(user), **g**(group), **o**(others/all). Tipurile de permisiuni sunt prescurtate in **r**(read),**w**(write),**x**(execute). Permiunile pot fi date sau luate prin **"+"** si **"-".** Tipurile de permisiuni pot fi copiate de la un grup la altul folosind **"="**. Exemplu:

```
#give execute permission to everyone on a file
chmod +x file_name
#remove execute permission for everyone on a file
chmod -x file_name
#give read permission to creator
chmod u+r file_name
#remove write permission for others (everyone but the creator and
groups that have this file assigned to them)
chmod o-w file_name
#copy creator permissions to group
chmod u=g file_name
```

2. **Acordati drepturile potrivite astfel incat oricine sa poata vizualiza continutul directoarelor abc si abc/t, sa poata adauga fisiere in abc/t, sa poata citi fisierele x, y, t1, t2, t3 din abc dar sa nu poata citi fisierele a si b din abc/t**

3. **Listati in format lung (*ls -l*) fisierele t, t1, t2, t3 din abc (sa se vada drepturile de acces asupra lui t, nu asupra fisierelor din el**

Procese

In Linux, comanda *cp /dev/zero /dev/null* produce un ciclu infinit. */dev/zero* produce un flux continuu de valori nule (zero), iar */dev/null* sterge tot inputul.

1. Executa comanda:

```
cp /dev/zero /dev/null
```

Aceasta comanda va porni un proces ce se va executa incontinuu. Puteti suspenda acest proces apasand **ctrl+z** sau puteti termina acest process apasand **ctrl+c**. Dupa ce este suspendat un process poate fi reactivat si dus in foreground cu comanda **fg**, sau reactivat si dus in background cu comanda **bg**. Procesele active se pot afisa cu comanda:

```
#list all processes started by user
ps -u
```

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
vlad       2256  0.0  0.2  31032  5404 pts/6    Ss+  17:58   0:00 bash
vlad       2609  0.0  0.0  18252   784 pts/6    T    18:18   0:00 less
vlad       2611  0.0  0.0  18252   884 pts/6    T    18:18   0:00 less
vlad       3030  0.0  0.0  22820   944 pts/6    T    19:05   0:00 grep --color=auto printf/usr/include
vlad       3692  0.0  0.2  30980  5196 pts/17   Ss   21:36   0:00 bash
vlad       3707 26.6  0.0  29208  1144 pts/6    R    21:37   1:02 cp /dev/zero /dev/null
```

In imaginea de mai sus, se poate vedea ca procesul pornit de comanda *cp /dev/zero /dev/null* are ca PID 3707. Folosind aces PID, putem termina procesul folosind comanda **kill**.

```
#remember to switch to your process' PID
kill 3707
```

2. Mutati procesul in background, afisati procesele active, terminati procesul pornit

Arhivare/Dezarhivare

In linux, comanda **tar** este folosita ca si utilitara primara pentru arhivare/dezarhivare.

Pentru a arhiva un director:

```
tar cvf arhive_name.tar dirname/
```

Parametrii reprezinta:

- c - creaza o noua arhiva
- v - listeaza toate fisierele procesate
- f - urmatorul parametru este numele arhivei

Pentru a dezarhiva:

```
tar xvf arhive_name.tar
```

Parametru nou reprezinta:

- x - extrage fisierele din arhiva

1. **Stergeti directoarele si refaceti structura pornind de la arhiva. Verificati conservarea datei crearii si a drepturilor de acces.**

Linkuri Simbolice (Symlinks)

In Linix, exista posibilitatea de a asocia o locatie fizica (director sau fisier) cu mai multe directoare/fisiere, ce pot avea nume diferite si se pot afla in locatii diferite. Comanda prin crea un symlink este:

```
ln -s /path/to/file /path/to/symlink1.
```

1. **Creati in tt un link simbolic cu numele c catre abc. Explorati functionarea lui, vis-a-vis de comenzile cd si pwd.**