

Activitat 01: Eliminació segura UNIX (AV1)

Professor: Jordi Mateo Fornés

Curs: Grau en Tècniques d'Interacció Digital i de Computació

Universitat: Universitat de Lleida - Campus Igualada

Estudiants: Isaac Brull i Eric Lopez

Pregunta 1: La vostra tasca és implementar en C aquesta funcionalitat.

El codi que hem implementat per crear la nostra “paperera” a UNIX, és el següent:



```
#include <sys/types.h>
#include <stdio.h>
#include <dirent.h>
#include <unistd.h>
#include <pwd.h>
#include <string.h>

int main(int argc, char* argv[]) {
    char current_dir[200];
    struct passwd *pw = getpwuid(getuid());
    const char *userHome = pw->pw_dir;
    char trashdir[] = "/.trash/";

    getcwd(current_dir, sizeof(current_dir));
    strcat(userHome, trashdir);

    if (mkdir(userHome, 0777) == 0)
        printf("directori .trash creat.\n");

    for (int i=1; i<argc; i++) {
        char arg[100];
        char aux_arg[200];
        strcpy(arg, current_dir);
        strcat(arg, "/");
        strcat(arg, argv[i]);
        strcpy(aux_arg, userHome);
        strcat(aux_arg, argv[i]);
        rename(arg, aux_arg);
    }
    return 0;
}
```

"rmsf.c" 32L, 774B 31,13 Bot

Les primeres línies de codi el que fan es poder utilitzar les rutes de l'arxiu i la carpeta trash si es que ja existeix, en el cas que no existeix encara la carpeta trash el que fa el següent “if” es crearla.

El bucle “for” és per a llegir cada arxiu que passa per argv, en cas que hi hagi més d'un. A l'hora de realitzar la comanda serà l'encarregat de moure l'arxiu seleccionat a la carpeta trash, això ho pot fer gràcies al “rename” de la última línia. Els “strings” auxiliars (arg i aux_arg), serveixen per anar afegint els arxius, i no modificar els strings originals, ja que si no el que faria el programa és copiar els mateixos directoris constantment a la carpeta .trash, en comptes de l'arxiu, dins dels directoris on treballem, és a dir:

/home/els8/.trash/home/els8/hola.txt

Pregunta 2: Expliqueu els passos realitzats

Pas 1 - Compilarem el codi i corregirem els errors que ens vagin sortint, per tal de disposar de l'executable que ens servirà per fer tests i comprovar que el funcionament del codi sigui l'esperat.

Pas 2 - Una vegada comprovat el funcionament del codi, el que farem és copiar l'executable a una direcció del PATH, per tal que el puguem executar a qualsevol directori del nostre sistema operatiu. El primer que hem fet, és canviar-nos a usuari "root", per tal de poder copiar un arxiu al PATH, i posteriorment hem executat la comanda `cp` a un directori del PATH, en aquest cas, el copiarem a la carpeta `/usr/bin`. La comandes que hem executat són les següents:

```
#su root
```

```
#cp rmsf /user/bin
```

Per comprovar que funciona tot, tornarem a canviar-nos al nostre usuari habitual.

Pregunta 3: La vostra tasca és preparar un fitxer test.sh que contingui totes les comandes que heu fet anar per testear la vostra solució.

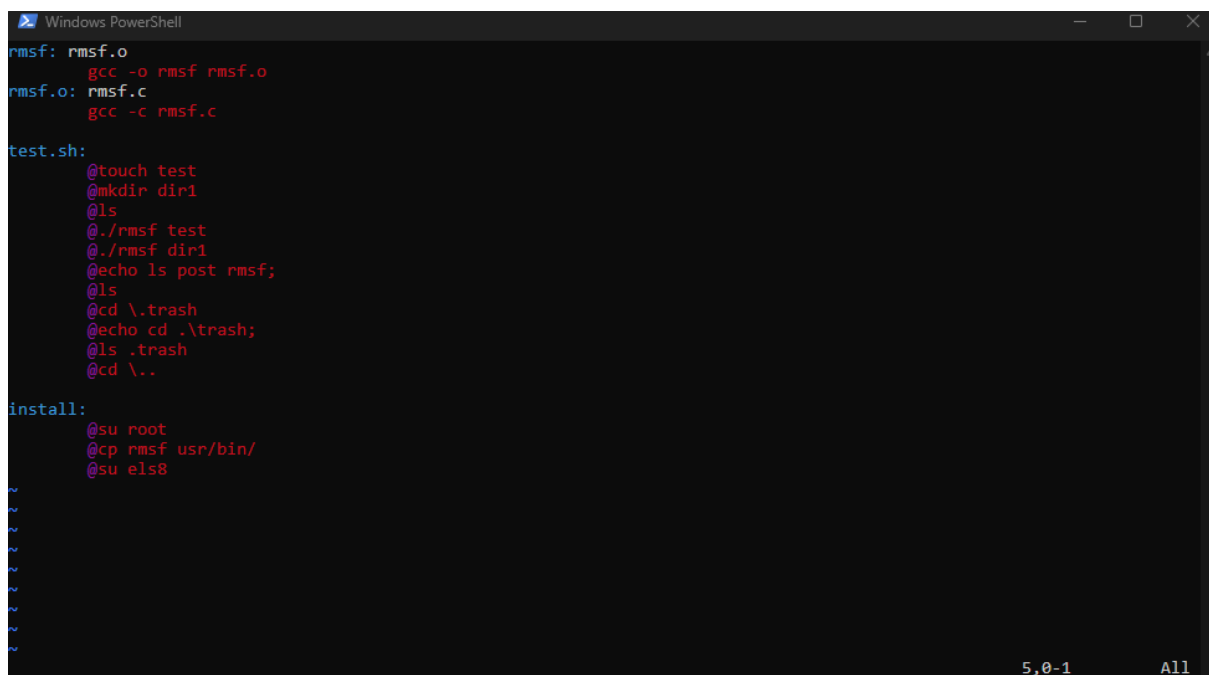
El nostre fitxer “test.sh” conté 11 comandes executades:

```
Windows PowerShell
@touch test
@touch test1
@mkdir dir1
@ls
@./rmsf test test1
@./rmsf dir1
@echo ls post rmsf;
@ls
@cd \.trash
@echo cd \.trash
@ls .trash
@cd \..
```

Tal com es pot veure, fem servir tres comandes `ls`, això ho fem servir per tal de buscar els arxius i directoris que hem creat, i si encara es troben al directori una vegada executar el programa. El tercer `ls` el fem per veure (una vegada dins de la carpeta `.trash`), si els arxius s'han mogut a la nostra "paperera".

Pregunta 4: La vostra tasca és revisar el funcionament dels Makefiles. Heu de preparar un Makefile que sigui capaç de compilar, executar, testejar amb el fitxer `test.sh` i instal·lar l'executable al sistema.

El makefile que hem implementat és el següent:



```
Windows PowerShell
rmsf: rmsf.o
gcc -o rmsf rmsf.o
rmsf.o: rmsf.c
gcc -c rmsf.c

test.sh:
@touch test
@mkdir dir1
@ls
@./rmsf test
@./rmsf dir1
@echo ls post rmsf;
@ls
@cd \.trash
@echo cd \.trash;
@ls .trash
@cd \..

install:
@su root
@cp rmsf usr/bin/
@su els8
```

Hem implementat un "test.sh" per poder executar totes les comandes que farem per comprovar que el codi compilat sigui correcte, ja que desconexem la forma d'executar un test.sh extern des del makefile.

El nostre "make install" fa exactament el que hem explicat anteriorment, canviant-nos a usuari root, executant la comanda, i tornant al nostre usuari.