

CRIDES AL SISTEMA OPERATIU DES DE C

1r GED

Fonaments d'Informàtica

9/12/2024

Joel Torrens Espada i Mar Massanas Morató

https://github.com/marmassanas/pr4_fonaments_informatica.git

Programa 1: my_tee.c

Objectiu:

my_tee és una implementació bàsica de la comanda tee en Linux. Llegeix dades des de l'entrada estàndard (stdin), les escriu simultàniament a la sortida estàndard (stdout) i les guarda en un arxiu especificat.

Funcionament pas a pas:

1. Arguments del programa: El programa rep el nom d'un arxiu com a argument. Si no es proporciona, mostra un missatge d'ús i acaba l'execució.

```
int main(int argc, char *argv[]) {  
    if (argc < 2) { //Comprova que el programa ha rebut com a mínim un argument (a part del nom del programa). Si no és així, mostra un missatge d'error  
        exit(-1);  
    }
```

2. Creació d'arxiu: Obre l'arxiu especificat en mode escriptura, creant-lo si no existeix o sobreescrivint-lo si ja hi és. Si no es pot obrir, mostra un error i surt del programa.

```
// Intenta crear un arxiu amb el nom especificat com a primer argument (argv[1]).  
int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, PERMISSIONS); // O_WRONLY: Obrir l'arxiu en mode només escriptura. O_CREAT: Crear l'arxiu si no existeix.  
//O_TRUNC: Si l'arxiu ja existeix, es buida.  
if (fd == -1) { //Si open falla (per exemple, per manca de permisos o espai insuficient), retorna -1  
    perror("Error creant l'arxiu");  
    exit(-1);  
}
```

3. Lectura i escriptura: Llegeix dades de l'entrada estàndard byte a byte (caràcter per caràcter). Escriu aquestes dades simultàniament a:

- La sortida estàndard (stdout) perquè l'usuari les pugui veure.
- L'arxiu especificat, per guardar-les permanentment.

Si hi ha errors en l'escriptura, el programa mostra un missatge i surt.

```
// Llegir caràcters d'entrada estàndard  
//Llegeix un caràcter (1 byte) de l'entrada estàndard (STDIN_FILENO) i l'emmagatzema a buffer.  
while ((bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer))) > 0) { //read retorna el nombre de bytes llegits  
  
    ssize_t stdout_written = write(STDOUT_FILENO, buffer, bytes_read); //envia el contingut llegit (emmagatzemat a buffer) directament a la terminal  
    if (stdout_written == -1) { // Comprova si 'write' a STDOUT falla  
        perror("Error escrivint a la sortida estàndard");  
        close(fd);  
        exit(-1);  
    }  
  
    // Escriu el contingut de buffer a l'arxiu indicat pel descriptor fd.  
    bytes_written = write(fd, buffer, bytes_read);  
    if (bytes_written == -1) { //Comprova si write falla (-1)  
        perror("Error escrivint a l'arxiu");  
        close(fd);  
        exit(-1);  
    }  
}
```

4. Finalització: Si l'entrada es tanca (usant Ctrl+D en la terminal), el programa finalitza i tanca l'arxiu.

Exemple d'execució:

1. Executar el programa amb un arxiu de destí:

./my_tee my_tee_prova.txt

2. Escriuiu text:

Hello World!

- Apareix a la terminal (sortida estàndard) i es guarda a sortida.txt.

3. Premeu Ctrl+D per acabar.

```
[(base) marmassanas@MacBook-Air-de-Mar pr4 % ./my_tee my_tee_prova.txt  
Hello World!Hello World!
```



my_tee_prova.txt
Hello World!

Programa 2: my_cmp.c

Objectiu:

my_cmp és una implementació bàsica de la comanda cmp en Linux. Compara dos arxius byte a byte per trobar diferències i indica:

- La posició del byte on es troba la primera diferència.
- La línia en què aquesta diferència es troba.

Funcionament pas a pas:

1. Arguments del programa: El programa rep dos noms d'arxiu com a arguments. Si no es proporcionen, mostra un missatge d'ús i acaba l'execució.

```
int main(int argc, char *argv[]) {  
    if (argc < 3) { //Comprova que s'han passat almenys dos arguments a la línia d'ordres (els noms dels fitxers).  
        fprintf(stderr, "Ús: %s <arxiu1> <arxiu2>\n", argv[0]);  
        exit(-1);  
    }  
}
```

2. Obertura d'arxius: Obre els dos arxius en mode lectura. Si algun dels arxius no es pot obrir, mostra un error i finalitza.

```
int fd1 = open(argv[1], O_RDONLY); //Obre el primer fitxer en mode només lectura  
if (fd1 == -1) { //Comprova si hi ha hagut un error en obrir el primer fitxer.  
    perror("Error obrint el primer arxiu");  
    exit(-1);  
}  
  
int fd2 = open(argv[2], O_RDONLY); //Obre el segon fitxer en mode només lectura  
if (fd2 == -1) { //Comprova si hi ha hagut un error en obrir el segon fitxer.  
    perror("Error obrint el segon arxiu");  
    close(fd1);  
    exit(-1);  
}
```

3. Comparació byte a byte: Llegeix un byte de cada arxiu i els compara:

- Si són diferents, mostra la posició del byte i el número de línia.
- Si un arxiu té més bytes que l'altre, indica que les longituds són diferents.

- Si els bytes són iguals, incrementa la posició del byte i el comptador de línies quan es troba un salt de línia (\n).

```
// Comparar byte a byte
while (1) { //Llegeix 1 byte de cada arxiu
    bytes_read1 = read(fd1, &buf1, 1); //Llegeix 1 byte del primer fitxer i l'emmagatzema a buf1. Retorna el nombre de bytes llegits (normalment 1 si tot va bé, 0 si
    bytes_read2 = read(fd2, &buf2, 1); //Llegeix 1 byte del segon fitxer i l'emmagatzema a buf2. Retorna el nombre de bytes llegits (normalment 1 si tot va bé, 0 si es

    // Comprovar si tots dos arxius han arribat al final
    if (bytes_read1 == 0 && bytes_read2 == 0) {
        // Són iguals si cap arxiu té bytes restants
        break;
    }

    // Comparar els continguts dels fitxers
    if (bytes_read1 != bytes_read2 || buf1 != buf2) {
        if (bytes_read1 != bytes_read2) { // Si un arxiu té més bytes que l'altre
            printf("Els arxius tenen diferents longituds.\n");
        } else { // Comprovar si hi ha una diferència
            printf("Diferència trobada al byte %d, línia %d\n", byte_position, line_number);
            //Comprova si els bytes actuals dels dos fitxers són diferents:Si són diferents, mostra un missatge amb la posició del byte i el número de línia.
        }
        close(fd1);
        close(fd2);
        return 0;
    }
    //Tanca els fitxers (close(fd1); close(fd2);) i surt del programa (return 0)
}

// Si el byte llegit és un salt de línia (\n), incrementa el comptador de línies.
if (buf1 == '\n') {
    line_number++;
}

byte_position++; //Incrementa la posició del byte processat
}
```

4. Finalització: Si s'ha arribat al final dels dos arxius sense trobar diferències, tanca els arxius i finalitza.

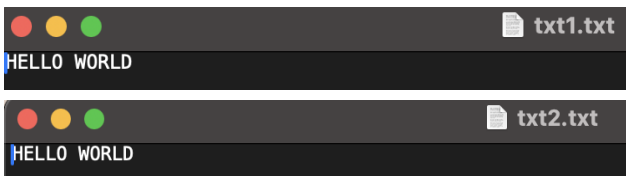
```
//Tanca els descriptors dels fitxers per alliberar recursos.
close(fd1);
close(fd2);
return 0; //Retorna 0 per indicar que el programa ha acabat correctament (a menys que ja hagi trobat una diferència).
```

Exemple d'execució:

1. Comparar dos arxius iguals:

./my_cmp txt1.txt txt2.txt

Sortida: -



```
(base) marmassanas@MacBook-Air-de-Mar pr4 % ./my_cmp txt1.txt txt2.txt
(base) marmassanas@MacBook-Air-de-Mar pr4 %
```

2. Comparar dos arxius diferents:

./my_cmp txt1.txt txt3t.txt

Sortida: Diferència trobada al byte n, línia y.

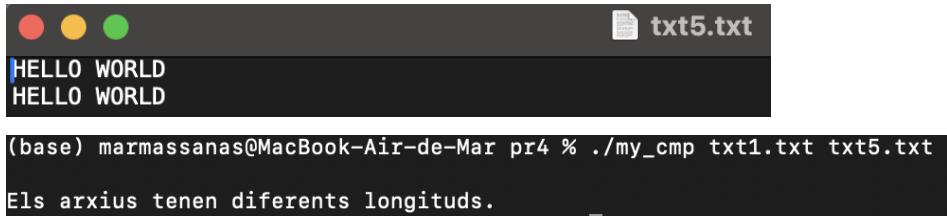


```
(base) marmassanas@MacBook-Air-de-Mar pr4 % ./my_cmp txt1.txt txt3.txt
Diferència trobada al byte 0, línia 1
```

3. Comparar arxius de longituds diferents:

./my_cmp txt1.txt txt5.txt

Sortida: Els arxius tenen diferents longituds.



The image shows two overlapping windows from a macOS environment. The top window is a file viewer titled 'txt5.txt' with a document icon. It displays the text 'HELLO WORLD' on two separate lines. The bottom window is a terminal window. The prompt is '(base) marmassanas@MacBook-Air-de-Mar pr4 %'. The command entered is './my_cmp txt1.txt txt5.txt'. The output of the command is 'Els arxius tenen diferents longituds.'

```
(base) marmassanas@MacBook-Air-de-Mar pr4 % ./my_cmp txt1.txt txt5.txt  
Els arxius tenen diferents longituds.
```