Grado en Ingeniería Informática

UNIVERSIDAD DE GRANADA

Resolución de examen
Módulo II

Sistemas Operativos

Autor:
Guillermo Bueno Vargas

Índice

1. Enunciado

NO ESTA PERMITIDO UTILIZAR CALCULADORA, MOVILES NI NIGÚN DISPOSITIVO DE COMUNICACION O MEDIO DE COMUNICACION. UNICAMENTE SE ACCEDERÁ A PRADO PARA ENTREGAR EL EXAMEN

Acceda a su cuenta personal especificando el código **`examenubu16`** y levantando **Ubuntu 16.04 (OJO: NO el de 64 bits)**

PARA ENTREGAR:

`Suba los archivos fuentes al trabajo definido en prado para esta prueba.`

`Importante: en la primera línea, como comentario, escriba apellidos+nombre`

Importante:

* use únicamente las llamadas al sistema estudiadas en los guiones de prácticas del presente módulo.
* los programas deben contener únicamente lo que se pide, se penalizará si incluyen funcionalidad no requerida.
* programe la actuación adecuada ante las distintas situaciones de error que puedan ocurrir.

Nota sobre tamaños de cadenas: incluya `<limits.h>` donde se define lo siguiente:
NAME_MAX = 14 (máximo nº de bytes en un nombre de archivo, exceptuando el caracter de terminacion nulo)
PATH_MAX = 1024 (máximo nº de bytes en una ruta, incluyendo el caracter de terminacion nulo)

EJERCICIO Variante 1

Construya un programa en C llamado `path1` que tendrá dos argumentos:
argv[1] el nombre de una orden (sin barras, por ejemplo `ls`)
argv[2] es un nº

El objetivo es averiguar si argv[1] está en el directorio número argv[2] de la lista de búsqueda, y en caso afirmativo mostrar en la salida estándar su tamaño en bytes, identificativo de usuario propietario e identificativo de grupo propietario.

El programa `path1` recoge el valor de $PATH de esta forma:

```
char *lista=getenv("PATH");
```

Debe escribir ese contenido en un archivo de nombre "listabusqueda" en el directorio de trabajo.

Creará un proceso hijo que ejecutará

```
cut -d: -f argv[2] –z listabusqueda
```

El proceso hijo se comunicará con el padre con un cauce (con nombre o sin nombre, a elegir). Si se opta por utilizar un cauce sin nombre se declarará `int pcut[2]` para tratar el cauce; si se opta por utilizar un archivo fifo se utilizará como descriptor el entero `int pcut`.

El proceso padre leerá del cauce en una única lectura una cadena

```
char rutadir[PATH_MAX];
```

Si el proceso hijo ha terminado con éxito y el nº de bytes leídos es 1 entonces la orden cut no ha encontrado el campo buscado, y ha escrito únicamente un carácter nulo en el cauce.
Si el proceso hijo ha terminado con éxito y el nº de bytes leídos es >1 entonces tenemos en rutadir la ruta número argv[2] de la lista de búsqueda.

Se averiguará `si argv[1]` existe dentro de `rutadir` y en caso afirmativo se mostrará en la salida estándar su tamaño en bytes, identificativo de usuario propietario e identificativo de grupo propietario.

EJERCICIO Variante 2

Construya un programa en C llamado `path` que tendrá un argumento:
argv[1] el nombre de una orden (sin barras, por ejemplo `ls`)

Realiza la función expresada en la variante 1 para todos los campos de la lista de búsqueda.

2. Código

```
1  //BUENO VARGAS, GUILLERMO
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <sys/wait.h>
6  #include <fcntl.h>
7  #include <limits.h>
8  #include <dirent.h>
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <errno.h>
12 #include <string.h>
13
14 int main (int argc, char* argv[])
15 {
16   int *pcut[2];
17   int *FD;
18   int fileout;
19
20   char *lista=getenv("PATH");
21
22   //LANZAMOS EL HIJO PARA CREAR LISTABUSQUEDA
23   int id;
24   if( (id = fork()) == -1 ){
25     perror("Error en fork()");
26     exit(EXIT_FAILURE);
27   }
28   else if (!id){
29     int leidos = strlen(lista);
30     fileout=open("listabusqueda", O_CREAT|O_TRUNC|O_WRONLY,S_IRUSR|S_IWUSR);
31     if (fileout < 0)
32     {
33       printf("El fichero de salida no se pudo abrir correctamente\n");
34       exit(-1);
35     }
36     write(fileout, lista, leidos);
37     close(fileout);
38   }
39
40   //ESPERAMOS AL HIJO, ASI NOS ASEGURAMOS QUE EL ARCHIVO LISTABUSQUEDA SE HA CREADO
41   int status_2;
42   if( waitpid(id, &status_2, 0) < 0){
43         perror("wait");
44   }
45
46
47   //EL REDIMENSIONADO
48   for (int i = 0; i < 2; i++){
49     pcut[i] = malloc( 2 * sizeof(int));
50     pipe (pcut[i]);
51   }
52
53   FD = pcut[0];
54
```

```
55
56  //LANZAMOS EL SEGUNDO HIJO PARA QUE EJECUTE CUT
57  int id_2;
58  if( (id_2 = fork()) == -1 ){
59    perror("Error en fork()");
60    exit(EXIT_FAILURE);
61  }
62  else if (!id_2){
63    close(FD[0]); //cerramos el descriptor de lectura en el cauce
64    close(STDOUT_FILENO);
65
66    //Establecemos la salida estAndar del proceso al descriptor de escritura del cauce
67    dup2(FD[1],STDOUT_FILENO);
68
69    if (execlp("cut","cut","-d:","-f",argv[2],"-z","listabusqueda",NULL) == -1){
70      perror("Error en execlp()");
71      exit (EXIT_FAILURE);
72    }
73
74  }
75
76  else{
77    close(FD[1]);
78    char rutadir[PATH_MAX];
79    int status;
80
81    if ( read(FD[0],rutadir,PATH_MAX) == -1){
82      perror("Error en read()");
83      exit(EXIT_FAILURE);
84    }
85
86    //CONCATENAMOS LA CADENA
87    strcat(rutadir,"/");
88    strcat(rutadir,argv[1]);
89
90
91    struct stat atributos;
92
93
94    if(stat(rutadir,&atributos) >= 0){
95      printf("\nDirectorio encontrado: %d bytes, %u id_usuario_propietario, %u ↩
           id_grupo_propietario \n", atributos.st_size, atributos.st_uid, atributos.st_gid);
96    }
97
98
99    int id_3;
100
101   if( waitpid(id_3, &status_2, 0) < 0 ) {
102           perror("wait");
103           exit(254);
104             }
105
106   //Si ls devuelve 0 indica que no esta en el directorio
107   if(WIFEXITED(status_2) == 0){
108           printf("Proceso %d devolvio %d\n", id_2, WEXITSTATUS(status));
109   }
```

```
110
111
112     if( waitpid(id_2, &status, 0) < 0 ) {
113             perror("wait");
114             exit(254);
115               }
116
117     if(WIFEXITED(status)){
118             printf("Proceso %d devolvio %d\n", id_2, WEXITSTATUS(status));
119     }
120
121   }
122
123
124   return 0;
125 }
```