SIMLAB2

Para entregar

Tienes que entregar un único fichero tar.gz con todos los ficheros fuente y el Makefile. Para crear el fichero tar puedes ejecutar desde el directorio donde tienes los ficheros el siguiente comando: %tar czvf simlab2.tgz *.c Makefile

Makefile

Crea un archivo *Makefile* para compilar todos los programas del examen, usando reglas independientes para cada uno. Incluye una regla llamada *clean* para eliminar todos los ejecutables, archivos objeto y/o ficheros temporales. Los programas deben compilarse si y sólo si se han actualizado los archivos de código fuente que los componen.

```
all: buscavocal sinvocales

buscavocal: buscavocal.c
    gcc -o buscavocal buscavocal.c

sinvocales: sinvocales.c
    gcc -o sinvocales sinvocales.c

clean:
    rm -r -f buscavocal sinvocales
```

1. buscavocal.c

Escribe un programa llamado buscavocal.c que reciba como parámetro el nombre de un fichero que debe existir y debe tener permiso de lectura y escritura. El programa debe crear una pipe con nombre, un fichero y un proceso hijo con el que se comunicará mediante la pipe con nombre.

La pipe con nombre se tiene que crear con permiso de lectura y escritura solo para el propietario de la pipe y con ningún permiso para el resto de usuarios de la máquina. El nombre de la pipe debe ser el nombre del fichero que ha recibido como parámetro añadiendo el sufijo ".pipe". Si la pipe existe debe dar un mensaje de aviso y continuar con la ejecución.

El fichero se tiene que llamar como el parámetro de entrada pero añadiendo el sufijo ".pos". Si el fichero ya existe deberá dar un error y acabar la ejecución. Los permisos de acceso tienen que ser de lectura y escritura para el propietario y solo de lectura para el resto de usuarios de la máquina.

El proceso hijo deberá leer todo el fichero que el usuario ha pasado como parámetro y escribir por la pipe con nombre todas las posiciones del fichero en las que ha encontrado una vocal. La escritura se tiene que hacer usando la representación interna del entero y sin ningún carácter adicional.

Por último, el proceso padre deberá leer de la pipe con nombre todas las posiciones y escribirlas en el fichero ".pos" en su representación interna, mostrar un mensaje por salida estándar indicando cuántas vocales se han encontrado y liberar el PBC de su hijo antes de acabar la ejecución. Ejemplo de ejecución:

```
%./buscavocal loremipsum
El fichero loremipsum contiene 1184 vocales
% ls -l loremipsum.*
prw------- 1 yolandab profes       0 Dec 16 11:01 loremipsum.pipe
-rw-r--r-- 1 yolandab profes    4736 Dec 16 11:01 loremipsum.pos
```

```c
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <errno.h>

void Usage(char *nom) {
char buf[80];
    sprintf(buf, "%s: parametros invalidos \n", nom);
    write(2, buf, strlen(buf));
    sprintf(buf, "%s Usage: %s fichero \n", nom, nom);
    write(2, buf, strlen(buf));
    exit(1);
}
void error_y_exit(char *msg){
    perror(msg);
    exit(1);
}
main(int argc,char *argv[]) {
int fd_pipe, fd_in, fd_pos, ret, pid1, pid2;
int nhijo=0, nchar=0, nvoc=0;
char c;
char buf[80];
if (argc !=2 ) {
    Usage(argv[0]);
}
if ((fd_in = open(argv[1], O_RDONLY)) < 0)  error_y_exit("abriendo fichero");
sprintf(buf,"%s.pos", argv[1]);
if ((fd_pos = open(buf, O_WRONLY|O_CREAT|O_EXCL, 0644)) < 0) error_y_exit("creando fichero");
sprintf(buf,"%s.pipe", argv[1]);
if (mknod(buf,S_IFIFO|0600,0) < 0){
    if (errno != EEXIST)    error_y_exit("creando la pipe");
    else write(2,"La pipe ya existia\n", strlen("La pipe ya existia\n"));
}
ret=fork();
if (ret == -1) {
    error_y_exit("creando primer hijo");
}
if (ret > 0) {
    if ((fd_pipe=open(buf,O_RDONLY)) < 0) error_y_exit("Error abriendo la pipe para leer");
    while ((ret = read(fd_pipe, &nchar, sizeof(nchar)))>0) {
        if (ret < sizeof(nchar)) {
            write(2, "Error de formato en los datos\n", strlen("Error de formato en los datos\n");
            exit(1);
        }
        write(fd_pos, &nchar, sizeof(nchar));
        nvoc++;
    }
    if (ret < 0) error_y_exit("leyendo de pipe");
    sprintf(buf, "El fichero %s contiene %d vocales\n", argv[1],nvoc);
    write(1, buf, strlen(buf));
    if (waitpid(-1, NULL, 0) <0) error_y_exit("Error en el waipid");
} else {
    if ((fd_pipe=open(buf,O_WRONLY))<0) error_y_exit("Error abriendo la pipe para escribir");
    while ((ret=read(fd_in,&c,sizeof(c))) > 0) {
        switch (c) {
        case 'a':
        case 'A':
        case 'e':
        case 'E':
        case 'i':
        case 'I':
        case 'o':
        case 'O':
        case 'u':
        case 'U': write(fd_pipe, &nchar, sizeof(nchar));
                  break;
        }
        nchar++;
    }
    if (ret < 0) error_y_exit("Error leyendo el fichero parametro");

}
}
```

2. sinvocales.c

Escribe un programa llamado sinvocales.c que reciba como parámetro el nombre de un fichero. Este fichero se tiene que haber utilizado antes como parámetro del programa buscavocal y por lo tanto debe existir tanto ese fichero como el fichero ".pos" que genera buscavocal.

Este programa para cada entero guardado en el fichero ".pos" debe comprobar primero que su valor es menor que el tamaño del fichero. En caso de que no lo sea debe acabar mostrando un mensaje de error por terminal. Si el entero es una posición válida dentro del fichero entonces debe acceder directamente a esa posición y escribir el carácter "*".

NOTA: junto con este enunciado os adjuntamos el fichero loremipsum junto con su derivado ".pos" para que podáis usarlo como entrada correcta de sinvocales (todos los enteros que contiene ".pos" son menores que el tamaño de loremipsum). También os adjuntamos el fichero loreimpsumfailed junto con su derivado ".pos" para que lo podáis usar como entrada incorrecta (contiene valores en ".pos" que no están dentro del fichero loreimpsumfailed)

Ejemplo de salida:

```
%./sinvocales loremipsum

% head -n 1 loremipsum
L*r*m *ps*m *d*r *m*t, c*ns*ct*t**r *d*p*sc*ng *l*t. *n*n t*rt*r
sc*l*r*sq** f*m*s r*s*s. M*ll*s s*m t*
mp*s *nt* n*str* **ct*r, p*lv*n*r c*ng**. **ct*r *l*q**m *ff*c*t*r
*ll*mc*rp*r *g*st*s n*ll*m *l*m*nt*m
 s*ll*c*t*d*n. P*s**r* f**g**t p*r*s f*sc* v*st*b*l*m *l*t, *r*t f*sc*
s*d*l*s. D*n*c *m*t m*x*m*s m*tt
*s c*nv*ll*s *x n*str* l**r**t *t. N*nc n*t*s **n**n *cc*ms*n l**r**t
n*s* m*gn* *rc* c*mm*d*.

%./sinvocales loremipsumfailed
Posición incorrecta: 2454 es mayor que el tamanyo (2452)

% head -n 1 loremipsumfailed
*r*m *ps*m *d*r *m*t, c*ns*ct*t**r *d*p*sc*ng *l*t. *n*n t*rt*r
sc*l*r*sq** f*m*s r*s*s. M*ll*s s*m t*
mp*s *nt* n*str* **ct*r, p*lv*n*r c*ng**. **ct*r *l*q**m *ff*c*t*r
*ll*mc*rp*r *g*st*s n*ll*m *l*m*nt*m
 s*ll*c*t*d*n. P*s**r* f**g**t p*r*s f*sc* v*st*b*l*m *l*t, *r*t f*sc*
s*d*l*s. D*n*c *m*t m*x*m*s m*tt
*s c*nv*ll*s *x n*str* l**r**t *t. N*nc n*t*s **n**n *cc*ms*n l**r**t
n*s* m*gn* *rc* c*mm*d*.
```

```c
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <stdio.h>

void Usage(char *nom) {
char buf[80];

    sprintf(buf, "%s: parametros invalidos \n", nom);
    write(2, buf, strlen(buf));
    sprintf(buf, "%s Usage: %s fichero \n", nom, nom);
    write(2, buf, strlen(buf));
    exit(1);
}

void error_y_exit(char *msg){
    perror(msg);
    exit(1);
}

main(int argc,char *argv[]) {
int fdv,fdp, ret, mida, pos;
char buf[160];

if (argc !=2 ) {
    Usage(argv[0]);
}

fdv = open(argv[1], O_WRONLY);
if (fdv < 0) error_y_exit("abriendo fichero parametro");
mida=lseek(fdv,0,SEEK_END);
if (mida < 0) error_y_exit("calculando tamanyo fichero");
sprintf(buf,"%s.pos",argv[1]);
fdp = open(buf, O_RDONLY);
if (fdp < 0) error_y_exit("abriendo fichero race");

while ((ret = read(fdp, &pos, sizeof(pos)))> 0) {
    if (ret < sizeof(pos)) {
        sprintf(buf, "Error en el formato del fichero pos");
        write(2,buf,strlen(buf));
        exit (1);
    }
    if (pos > mida) {
        sprintf(buf, "Posición incorrecta: %d es mayor que el tamanyo (%d)\n",pos, mida);
        write(2,buf,strlen(buf));
        exit (1);
    }
    if (lseek (fdv,pos,SEEK_SET) < 0) error_y_exit("Error en el lseek");
    write(fdv,"*",sizeof(char));
}
if (ret < 0) error_y_exit("Error en el read de fichero");


}
```