Sistemas Operativos

Formulario de auto-evaluación

Modulo 2. Sesión 6. Control de archivos y archivos proyectados en memoria

Nombre y apellidos:

Néstor Rodríguez Vico

a) Cuestionario de actitud frente al trabajo.

El tiempo que he dedicado a la preparación de la sesión antes de asistir al laboratorio ha sido de ... 45... minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: ...Si... (si/no). En caso de haber contestado “no”, indica los motivos por los que no las has resuelto:

2. Tengo que trabajar algo más los conceptos sobre:

3. Comentarios y sugerencias:

b) Cuestionario de conocimientos adquiridos.

Mi solución a la **ejercicio 1** ha sido:

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <stdbool.h>

int main(int argc, char *argv[]) {
	if(argc != 4) {
		printf("Modo de uso: %s <programa> <simbolo> <archivo>\n\n", argv[0]);
		exit(1);
	} else {
		char *comando, *archivo;
		int fd;

		comando = argv[1];
		archivo = argv[3];

		if (strcmp(argv[2], "<") == 0) {
			//entrada
			fd = open (archivo, O_RDONLY);
			close(STDIN_FILENO);
			if (fcntl(fd, F_DUPFD, STDIN_FILENO) == -1)
				perror ("Error en fcntl");
		} else if (strcmp(argv[2], ">") == 0) {
			//salida
			fd = open (archivo, O_CREAT|O_WRONLY);
			close (STDOUT_FILENO);
			if (fcntl(fd, F_DUPFD, STDOUT_FILENO) == -1)
				perror ("Error en fcntl");
		} else {
			printf("Debe pasarse \"<\" o \">\" con las comillas %s\n\n", argv[2]);
			exit(1);
		}

	   if((execlp(comando, "", NULL) < 0)) {
			perror("Error en el execlp\n");
			exit(-1);
		}
```

```
                close(fd);
        }
}
```

Mi solución a la **ejercicio 3** ha sido:

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>

int main(int argc, char *argv[]) {
        struct flock cerrojo;
        int fd, i;

        if(argc != 2) {
                printf("Modo de uso: %s <archivo>\n", argv[0]);
                exit(1);
        } else {
                //Nombre del archivo
                char *archivo = argv[1];

                //Abrimos el archivo
                if ((fd=open(archivo, O_RDWR)) == -1 ){
                        perror("Fallo al abrir el archivo");
                        return 0;
                }

                cerrojo.l_type=F_WRLCK;
                cerrojo.l_whence=SEEK_SET;
                cerrojo.l_start=0;
                //Bloquear archivo entero
                cerrojo.l_len=0;

                //Intentamos un bloqueo de escritura del archivo
                printf ("Intentando bloquear %s\n", archivo);
                if (fcntl (fd, F_SETLKW, &cerrojo) == EDEADLK) {
                        //Si el cerrojo falla, pintamos un mensaje
                        printf ("El cerrojo ha fallado");
```

```
                }

                //El bloqueo tiene exito -> procesar el archivo
                printf ("Procesando el archivo %s\n", archivo);

                //sleep para lanzar otra vez el programa
                for (i = 0; i < 10; i++) {
                        sleep(1);
                }

                //Desbloqueamos el archivo
                cerrojo.l_type=F_UNLCK;
                cerrojo.l_whence=SEEK_SET;
                cerrojo.l_start=0;
                cerrojo.l_len=0;
                if (fcntl (fd, F_SETLKW, &cerrojo) == -1) {
                        perror ("Error al desbloquear");
                }

                return 0;
        }
}
```

Mi solución a la **ejercicio 5** ha sido:

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>

int main(int argc, char *argv[]) {
        if(argc != 3) {
                printf("Modo de uso: %s <origen> <destino>\n\n", argv[0]);
                exit(1);
        } else {
```

```
struct stat sb;
		char *origen = argv[1], *destino = argv[2];
		int fd_origen, fd_destino, tama;
		char *mem_orig, *mem_dest;

		//Abrimos el fichero de origen
		fd_origen = open(origen, O_RDONLY);
		if (fd_origen == -1) {
			perror("Fallo al abrir el archivo de origen\n");
			exit(1);
		}

		//Obtenemos su stat
if (fstat (fd_origen, &sb) == -1) {
	printf("Error al hacer stat\n");
	exit(1);
}

if (!S_ISREG (sb.st_mode)) {
	printf ("El fichero no es regular\n");
	exit(1);
}

//Guardamos el tamaño
tama = sb.st_size;

//Creamos el archivo de destino
umask(0);
		fd_destino = open(destino, O_RDWR|O_CREAT|O_EXCL, S_IRWXU);
		if (fd_destino == -1) {
			perror("Fallo al crear el archivo");
			exit(1);
		}

		//Asignamos el espacio en el destino
		ftruncate(fd_destino, tama);

		//Creamos el mapa de memoria de origen
		mem_orig = (char *) mmap(0, tama, PROT_READ, MAP_SHARED, fd_origen, 0);
		if(mem_orig == MAP_FAILED) {
			perror("Fallo en mmap (origen)");
			exit(1);
		}
```

```
		//Creamos el mapa de memoria de destino
		mem_dest = (char *) mmap(0, tama, PROT_WRITE, MAP_SHARED, fd_destino, 0);
		if(mem_dest == MAP_FAILED) {
			perror("Fallo en mmap (destino)");
			exit(1);
		}

		memcpy(mem_dest, mem_orig, tama);
		munmap(mem_orig, tama);
		munmap(mem_dest, tama);
		close(fd_origen);
		close(fd_destino);

		return 0;
	}
}
```