

# Gestión básica de procesos

## Definitions.h

En este .c será el mismo que el apartado de la practica 1.1, sin embargo, cambiamos en la estructura dentro de los nodos que, en vez de tener el acceso al siguiente nodo y un valor entero, tendrá el acceso al siguiente, pero cambia el valor que será una cadena char y también tendrá el pid.

Además, en la cabecera se define la ejecución del procesador y contador.

Y su función será el definir los métodos de la lista.c.

## Lista.c

En este .c para guardar cadenas en la lista, se hará un strcpy en el valor de la lista, en todos los casos de insertar y en el del getElementoN, pero en este último solo haría falta un return que devuelva el valor en esa posición de coincidencia.

En este apartado me surgieron un par de problemas ya que no estuve prestando mucha atención en la clase de presentación, ya que dijiste que para guardar una cadena haría falta un strcpy, sin embargo, yo solo atendí justo en la parte que había que utilizar esa función, pero no para qué. Por ello en primera opción para guardar el valor de la cadena, utilicé un malloc de carácter char para guardar en el nodo pAux1->valor a esa dirección de memoria, pero no fue así, ya más tarde pregunté a un compañero y ya me dijo que había que utilizar ese strcpy para guardar el valor en el pAux1->valor.

## Manager.c

En este .c solo haría falta entender cómo funciona el código, por lo que comentaré su función de una manera más general.

Al principio se encuentra la definición de métodos a ejecutar, y la estructura de los patrones, más tarde en el main se ejecuta algunos métodos como crear la lista patrones ya dada en la lista.c, y el manejador de señales y luego ya está el hecho de hacer la gestión de procesos.

Cabe destacar que antes de la gestión de proceso, recoge de la línea de ordenes los ficheros a utilizar, para leer solo el documento en el que se encuentra el texto y saber cuántas líneas tiene.

En esta gestión de procesos, primero crea un método en el que lee el texto de patrones y lee los datos en forma de líneas (fgets), después esa línea la descompone hasta encontrar un espacio entre palabras gracias a la función strtok que su función es recoger un string hasta un carácter específico que en este caso es el espacio. Y después cada palabra encontrada la inserta en la lista.

Después tenemos la creación de procesos, el cuál lee el fichero de texto, y con un while ejecuta los procesos contadores y con el for crea los procesos procesadores. En este apartado en el caso del while recoge cada línea y se la manda al contador.c y este hace que cuente las palabras. Y en el for crea desde la posición 2 hasta la longitud de esta lista, y para cada proceso del procesador manda un patrón diferente de la lista utilizando el método getElementoN, que recoge cada patrón.

Cabe destacar que en cada proceso creado se le atribuye el pid del proceso creado para luego mostrarlo por terminal.

En cada llamada a la creación de proceso después la guarda en el array de la tabla. Cabe señalar que como el valor es una cadena de carácter array en las líneas 222 y 199 se hace un strcpy en vez de una asignación normal.

Y para finalizar hace los métodos de esperar procesos con el wait(), y utiliza el método destruir para liberar recursos y más tarde el hecho de terminar los procesos con la señal kill.

## Contador.c

En este .c ya estaba casi hecho, solo haría falta hacer la llamada en el main con el inconveniente que hace falta pasar el argumento referido al número de la línea a entero.

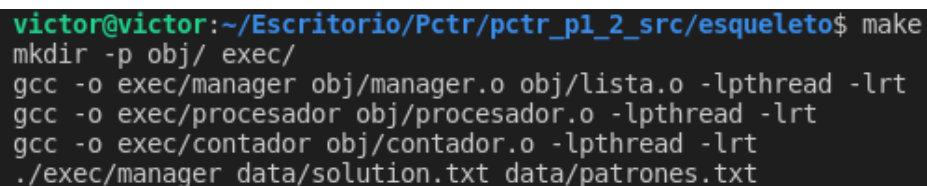
Su función se basa en el que le llega la línea por parámetros y este hace que con un switch cuenta las palabras dentro de esa línea.

## Procesador.c

Y por último este .c, para ello primero leeríamos cada línea del fichero de texto con fgets, y utilizaríamos la función strtok para delimitar cada palabra y así cada palabra compararla con el patrón en ese momento de ejecución.

## Compilar y ejecutar

Para compilar unos dirigiremos al esqueleto del proyecto, y pondremos en el comando de órdenes, "make", y se nos desplegará las siguientes líneas:

A terminal window with a dark background and light-colored text. The prompt is 'victor@victor:~/Escritorio/Pctr/pctr\_p1\_2\_src/esqueleto\$'. The commands entered are: 'make', 'mkdir -p obj/ exec/', 'gcc -o exec/manager obj/manager.o obj/lista.o -lpthread -lrt', 'gcc -o exec/procesador obj/procesador.o -lpthread -lrt', 'gcc -o exec/contador obj/contador.o -lpthread -lrt', and './exec/manager data/solution.txt data/patrones.txt'.

```
victor@victor:~/Escritorio/Pctr/pctr_p1_2_src/esqueleto$ make
mkdir -p obj/ exec/
gcc -o exec/manager obj/manager.o obj/lista.o -lpthread -lrt
gcc -o exec/procesador obj/procesador.o -lpthread -lrt
gcc -o exec/contador obj/contador.o -lpthread -lrt
./exec/manager data/solution.txt data/patrones.txt
```

El cual esta imagen te muestra todas las compilaciones que ha hecho, y para la ejecución del programa solo haría falta copiar la última línea mostrada y ejecutarla en el comando de órdenes.