Practica 1.2

Esta es la memoria de la práctica 1.2 de programación concurrente del grupo L6 formado por **Juan José García García, y Jaime Montalvo Alfageme.**

Para el desarrollo del código pedido, hemos llevado una metodología de trabajo bastante estructurada, primero estudiando el comportamiento del manager proporcionado.

Una vez que entendíamos el código a la perfección, hemos pensado cómo implementar el código.

Para esto hemos estudiado lo solicitado y hemos usado un método de abstracción, buscando resolver muchos problemas pequeños, de esta forma hemos encontrado métodos de librerías implementadas, como “strstr”, con esta forma de pensar conseguimos hacer un código muy simple, pero funcional y óptimo.

Para estructurar la lista enlazada, hemos cambiado los tipos de datos de int, a un puntero a un dirección de memoria, que al insertar un valor, la asignamos usando malloc, de esta forma podemos guardar la memoria necesaria, el parámetro que se le pasa a malloc, es el tamaño en memoria de la string a almacenar, esto se obtiene con strlen de la string, pero le sumamos uno para poder almacenar el valor nulo de terminación. De esta forma hemos podido adaptar toda la lista. Aqui un ejemplo de el método insertar:

```
// Inserta al ppio de la lista.
void insertar(TLista *pLista, char* valor)
{
   TNodo *pAux1 = malloc(sizeof(TNodo));
   pAux1->valor = malloc(strlen(valor)+1);
   strcpy(pAux1->valor, valor);
   pAux1->pSiguiente = pLista->pPrimero;
   pLista->pPrimero = pAux1;
}
```

Para los procesos hemos usado el mismo método de abstracción.

Para el procesador hemos abierto el archivo con fopen y los parámetros siendo el valor pasado en la ejecución, y el carácter r, de lectura.
Después usamos un bucle for:

```
for(int lin = 1; fgets(linea, sizeof(linea), archivo);lin++){
  linea[strcspn(linea, "\n")] = 0;
  if (strstr(linea, argv[2]) != NULL) {
    printf("[PROCESADOR %d] Patrón '%s' encontrado en línea %d\n", getpid(), argv[2], lin);
  }
}
```

Como se puede apreciar en la sentencia de comprobación no usamos una comparación con la variable asignada, si no que usamos el método fgets, que recibe todos los valores de un archivo hasta el final de la línea, luego asignamos al final un 0 para el final de la línea, y comprobamos con strstr, si existe el patrón en la línea, y si se encuentra imprimimos el

resultado usando la variable del for para el número de la línea. Después se cierra la lectura del archivo y se finaliza el proceso exitosamente. Siendo este el final del proceso procesador.

Para el proceso contador simplemente hemos tenido que llamar al método ofrecido y retornar la salida exitosa siendo un codigo super simple como se muestra a continuación:

```c
int main(int argc, char *argv[])
{
  contar(argv[1], atoi(argv[2]));

  return EXIT_SUCCESS;
}
```

Y con esto finaliza la práctica.

Para la ejecucion simplemente debemos usar el comando make all, dado que tambien hemos implementado el make clear en el make all, de esta forma no es necesario ejecutar los comandos por separado, y se limpia, compila, y ejecuta con solo un comando.

Para esto solo hemos añadido clean al principio de la sentencia de all:

```
all : clean dirs manager procesador contador test
```