SISTEMAS OPERATIVOS I - Práctica 2 (Parte 2)
Grado en Ingeniería Informática - Escuela Superior de Informática (UCLM))

1. Actividades de Laboratorio

Escriba un programa C estándar para cada uno de los distintos enunciados con la funcionalidad indicada. Salvo que se especifique lo contrario, se entenderá que la entrada y salida del programa corresponderá a la entrada estándar y salida estándar. Se aconseja utilizar la función `scanf` de la biblioteca estándar para la lectura de números enteros o reales de la entrada salvo otra indicación en el enunciado de la actividad.

1. Escriba una función llamada `leerlinea` que tenga dos argumentos, el primero de ellos será un vector de caracteres y el segundo argumento será un entero que indicará el número máximo de caracteres que puede almacenarse en el vector del primer argumento.

   La función debe leer una línea de la entrada estándar y almacenar sus caracteres en el primer argumento de la función. Supóngase que el tamaño máximo que puede tener una línea de la entrada es algo fijo y razonable.

   Además, se va a preparar un archivo de funciones, llamado `util.c`, que pueden resultar útiles en otros ejercicios posteriores. La función `leerlinea` debe formar parte de este archivo. Se debe compilar dicho archivo y obtener el módulo de código objeto `util.o`.

   Por último, escriba un programa que lea líneas de la entrada estándar y copie en la salida estándar únicamente la línea más larga que haya leído. Para ello utilice la función del módulo `util.o`

2. Incorpore una función llamada `miatoi` al archivo `util.c` con la funcionalidad de `atoi` para enteros positivos de la biblioteca estándar de C y escriba un programa que utilice dicha función

3. Dado el siguiente programa, indique de forma razonada cuántas zonas distintas de memoria utiliza el programa y la justificación del valor de las variables mostrado en la salida

```
#include <stdio.h>

int vg1;
int *vg2;

void funcion1(void) {
  int i;
  int x = 1, y = 2, z[3];
  int *ip;

  ip = &x;
  y = *ip;
  *ip = 0;
  ip = &z[0];
  for (i = 0; i < 3; i++)
    *ip++ = i;

  printf("\nVARIABLES DE funcion1\n");
  printf("   i - Dirección: %p - Valor: %d\n", (void *)&i, i);
  printf("   x - Dirección: %p - Valor: %d\n", (void *)&x, x);
  printf("   y - Dirección: %p - Valor: %d\n", (void *)&y, y);
  for (i=0; i<3; i++)
    printf("z[%d] - Dirección: %p - Valor: %d\n", i, (void *)&z[i], z[i]);
  printf("  ip - Dirección: %p - Valor: %p\n", (void *)&ip, (void *)ip);
}

void funcion2(void) {
  int vf2;
```

```
29
      printf("\nVARIABLES DE funcion2\n");
31    printf("vf2 - Dirección: %p - Valor: %d\n", (void *)&vf2, vf2);
  }
33  int main() {
    int vm1;
35
    printf("VARIABLES GLOBALES\n");
37  printf("vg1 - Dirección: %p - Valor: %d\n", (void *)&vg1, vg1);
    printf("vg2 - Dirección: %p - Valor: %p\n", (void *)&vg2, (void *)vg2);
39
    printf("\nVARIABLES DE main\n");
41  printf("vm1 - Dirección: %p - Valor: %d\n", (void *)&vm1, vm1);

43  printf("\nFUNCIONES\n");
    printf("funcion1 - Dirección %p\n", (void *)funcion1);
45  printf("funcion2 - Dirección %p\n", (void *)funcion2);
    printf("    main - Dirección %p\n", (void *)main);
47
    funcion1();
49  funcion2();

51  return 0;
  }
```

4. Incorpore una función llamada `mistrcpy` al archivo `util.c` con la funcionalidad de `strcpy` de la biblioteca estándar de C y escriba un programa que utilice dicha función

5. Incorpore al archivo `util.c` una función llamada `void reverse(char *s)` que invierta la cadena de caracteres `s` en la propia cadena `s`. Úsela para escribir un programa que invierta su entrada, línea a línea y se envíe a la salida

6. Incorpore al archivo `util.c` una función llamada `swap` que intercambie el valor de dos variables enteras que pueden estar definidas en cualquier punto del programa. Escriba un programa que utilice dicha función

7. Copie en una única línea de la salida estándar los argumentos de la línea de ordenes

8. Copiar el contenido de los ficheros indicados en la línea de ordenes en la salida estándar. Si no se especifica ningún argumento se debe copiar la entrada en la salida. Es decir, la funcionalidad del comando **cat** [<*archivo*>]*

9. La salida del programa será una copia de las líneas de la entrada que contengan un patrón indicado en su argumento. El patrón consistirá exclusivamente en una secuencia de caracteres sin ningún metacarácter. El programa debe tener dos opciones con la sintaxis habitual de Unix (con el carácter inicial `'-'` y en cualquier combinación), `'x'` para copiar las líneas que no contengan el patrón y `'n'` para indicar al inicio de cada línea de salida con el nº de línea de la entrada seguido del carácter `':'` y la línea de entrada. Es decir, una funcionalidad similar al comando `grep` con las opciones `'v'` y `'n'`

10. El programa debe calcular el valor máximo, el valor mínimo y la media aritmética de una serie de números enteros obtenidos de la entrada. La salida del programa serán tres líneas con el siguiente formato:

    ```
    Máximo: <valor máximo>
    Mínimo: <valor mínimo>
    Media: <valor medio>
    ```

Los valores máximo y mínimo deberán escribirse como enteros mientras que el valor medio se escribirá con dos cifras decimales

11. El programa debe calcular todas las raíces de polinomios de segundo grado. La entrada será una secuencia de líneas que contendrá una serie de números reales que, tomados de tres en tres, se interpretarán como los coeficientes a, b y c de cada polinomio de la forma ax^2+bx+c. La salida del programa debe consistir en una línea por cada polinomio leído con el siguiente formato:

    ```
    Raíces de ax^2+bx+c : x1 x2
    ```

    Si x1 o x2 son valores reales se escribirán con tres cifras decimales y si son números complejos de la forma R+Ii donde R e I se escribirán con tres cifras decimales

12. Construir un programa llamado **formato** que imprima en la salida estándar el contenido de una serie de archivos cuyos nombres se pasan en la línea de órdenes. El programa admitirá las siguientes opciones:

    ```
    -m indicará que la salida deberá escribirse toda en minúsculas
    -M indicará que la salida deberá escribirse toda en mayúsculas
    ```

    Se podrá usar a lo más una de las opciones anteriores. Si no se indica ninguna de las anteriores la salida deberá hacerse copiando la entrada como esté (mayúscula o minúscula). Adicionalmente podrá tenerse una opción -n, donde n es un entero mayor que cero. Si se usa esta opción, n será el interlineado que deba usarse (1 indica espacio simple, 2 doble espacio, etc.) Si no está presente la opción anterior se imprimirá a espacio simple. La sintaxis de la línea de órdenes será

    formato [-<n>] [-m] [-M] [<*archivo*>]$^+$

13. Construir un programa llamado **frecuencia** que imprima en la salida estándar el nº de veces que aparece las distintas palabras de la entrada que empiecen por un carácter alfabético y el nº de la última línea donde apareció cada palabra. Se considerarán como separadores de palabras los caracteres blancos, los tabuladores horizontales y los caracteres de nueva línea. El programa debe calcular la frecuencia de cualquier número de palabras de la entrada y usar la mínima memoria posible. El formato de la salida consistirá en una línea por palabra distinta y en el mismo orden que aparecen en la entrada con la siguiente estructura:

    ```
    <palabra> <frecuencia> <nº de la última línea>
    ```