# Práctica 1: Introducción a la programación de sistemas en Linux

## Índice

**1 Introducción** **1**
1.1 Objetivos . . . . . 1
1.2 Requisitos . . . . . 1

**2 Ejercicios** **2**
Ejercicio 1 . . . . . 2
Ejercicio 2 . . . . . 2
Ejercicio 3 . . . . . 3
Ejercicio 4 . . . . . 3
Ejercicio 5 . . . . . 5

## 1 Introducción

### 1.1 Objetivos

- Familiarizarse con el entorno de desarrollo de aplicaciones C en GNU/Linux, y comprender los conceptos de proyecto y ejecutable en este contexto.
- Familiarizarse con el manejo básico del shell y aprender a desarrollar *shell scripts* sencillos.

### 1.2 Requisitos

Para poder realizar con éxito la práctica el alumno debe haber leído y comprendido los siguientes documentos facilitados por el profesor:

- Presentación "Introducción al entorno de desarrollo", que nos introduce al entorno GNU/Linux que utilizaremos en el laboratorio, y describe cómo trabajar con proyectos C con Makefile.
- Presentación "Revisión: Programación en C", que realiza un repaso de los conocimientos de C necesarios para realizar con éxito las prácticas, haciendo especial hincapié en los errores que cometen habitualmente los estudiantes menos experimentados en el lenguaje C.
- Manual del laboratorio titulado "Entorno de desarrollo C para GNU/Linux", que describe las herramientas que componen el entorno de desarrollo que vamos a utilizar, así como las funciones básicas de la biblioteca estándar de C que los alumnos deben conocer.
- Presentación "Introducción a Bash", que presenta una breve introducción al interprete de órdenes (shell) Bash.

# 2 Ejercicios

## Ejercicio 1

Analizar el código del programa `show_file.c`, que lee byte a byte el contenido de un fichero, cuyo nombre se pasa como parámetro, y lo muestra por pantalla usando funciones de la biblioteca estándar de "C". Responda a las siguientes preguntas:

- ¿Qué comando se debe emplear para generar el ejecutable del programa (`show_file`) invocando directamente al compilador `gcc` (sin usar `make`)?
- Indique dos comandos para llevar a cabo respectivamente la compilación del programa (generación de fichero objeto) y el enlazado del mismo de forma independiente.

Realice las siguientes modificaciones en el programa `show_file.c`:

1. Realizar la lectura byte a byte del fichero de entrada empleando la función `fread()` en lugar de `getc()`. Modificar también la invocación a la función `putc()` por una llamada a `fwrite()`
2. Añadir un parámetro al programa modificado para permitir al usuario especificar el tamaño de bloque en bytes a usar en cada lectura realizada por `fread()`.

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    FILE* file=NULL;
    int c,ret;

    if (argc!=2) {
        fprintf(stderr,"Usage: %s <file_name>\n",argv[0]);
        exit(1);
    }

    /* Open file */
    if ((file = fopen(argv[1], "r")) == NULL)
        err(2,"The input file %s could not be opened",argv[1]);

    /* Read file byte by byte */
    while ((c = getc(file)) != EOF) {
        /* Print byte to stdout */
        ret=putc((unsigned char) c, stdout);

        if (ret==EOF){
            fclose(file);
            err(3,"putc() failed!!");
        }
    }

    fclose(file);
    return 0;
}
```

## Ejercicio 2

El programa *badsort-ptr.c*, cuyo código fuente se muestra a continuación, ha sido desarrollado para realizar una ordenación por el método de la burbuja aplicada a un *array* de pares (cadena de caracteres, entero) inicializado dentro del programa. El programa emplea aritmética de punteros para acceder a los distintos elementos del array durante el recorrido. Lamentablemente, el programador ha cometido algunos errores. Utilizando un depurador de C (p.ej.,`gdb`) el alumno debe encontrar y corregir los errores.

```c
#include <stdio.h>

typedef struct {
    char data[4096];
    int key;
} item;

item array[] = {
    {"bill", 3},
    {"neil", 4},
    {"john", 2},
    {"rick", 5},
    {"alex", 1},
};

void sort(item *a, int n) {
    int i = 0, j = 0;
    int s = 1;
    item* p;

    for(; i < n & s != 0; i++) {
        s = 0;
        p = a;
        j = n-1;
        do {
            if( p->key > (p+1)->key) {
                item t = *p;
                *p  = *(p+1);
                *(p+1) = t;
                s++;
            }
        } while ( --j >= 0 );
    }
}

int main() {
    int i;
    sort(array,5);
    for(i = 0; i < 5; i++)
        printf("array[%d] = {%s, %d}\n",
                i, array[i].data, array[i].key);
    return 0;
}
```

## Ejercicio 3

## Ejercicio 4

Estudiar el código y el funcionamiento del programa `show-passwd.c`, que lee el contenido del fichero del sistema `/etc/passwd` e imprime por pantalla (o en otro fichero dado) las distintas entradas de`/etc/passwd` –una por línea–, así como los distintos campos de cada entrada. El fichero `/etc/passwd` almacena en formato de texto plano información esencial de los usuarios del sistema, como su identificador numérico de usuario o grupo así como el programa configurado como intérprete de órdenes (*shell*) predeterminado para cada usuario. Para obtener más información sobre este fichero se ha de consultar su página de manual: `man 5 passwd`

El modo de uso del programa puede consultarse invocándolo con la opción -h:

```
$ ./show-passwd -h
Usage: ./show-passwd [ -h | -v | -p | -o <output_file> ]
```

Las opciones `-v` y `-p`, permiten configurar el formato en el que el programa imprime la información de `/etc/passwd`. Las citadas opciones activan respectivamente el modo `verbose`(por defecto) o `pipe`. La opción `-o`, que acepta un argumento obligatorio, permite selecionar un fichero para la salida del programa alternativo a la salida estándar.

Uno de los principales objetivos de este ejercicio es que el estudiante se familiarize con tres funciones muy útiles empleadas por el programa `show-passwd.c` , y cuya página de manual debe consultarse:

- `int sscanf(const char *s, const char *format, ...);`

Variante de `scanf()` que permite leer con formato a partir de un buffer de caracteres pasado como primer parámetro (`s`) . La función almacena en variables del programa, pasadas como argumento tras la cadena de formato, el resultado de convertir los distintos "tokens" de `s` de ASCII a binario.

- `char *strsep(char **stringp, const char *delim);`

Permite dividir una cadena de caracteres en *tokens*, proporcionando como segundo parámetro la cadena delimitadora de esos tokens. Como se puede observar en el programa `show-passwd.c`, esta función se utiliza para extraer los distintos campos almacenados en cada línea del fichero `/etc/passwd`, que están separados por `":"`. La función `strsep()` se usa típicamente en un bucle, que para tan pronto como el token devuelto es NULL. El primer argumento de la función es un puntero por referencia. Antes de comenzar el bucle, `*stringp` debe apuntar al comienzo de la cadena que deseamos procesar. Cuando `strsep()` retorna, `*stringp` apunta al resto de la cadena que queda por procesar.

- `int getopt(int argc, char *const argv[], const char *optstring);`

Esta función es la más sofisticada de las tres, y permite procesar cómodamente las distintas opciones de la línea de comando que acepta un programa C. La función suele invocarse desde `main()`, y sus dos primeros parámetros coinciden con los argumentos `argc` y `argv` pasados a `main()`. El parámetro `optstring` sirve para indicar de forma compacta a `getopt()` cuáles son las opciones que el programa acepta –cada una identificada por una letra–, y si éstas a su vez aceptan parámetros obligatorios u opcionales.

El estudiante deberá familiarizarse con esta función mediante el estudio del código fuente del programa, y la consulta de la página de manual de `getopt()`: `man 3 getopt`

Deben tenerse en cuenta las siguientes consideraciones:

1. La función `getopt()` se usa en combinación con un bucle, que invoca tantas veces la función como opciones ha pasado el usuario en la línea de comandos. Cada vez que la función se invoca y encuentra una opción, `getopt()` retorna el caracter correspondiente a dicha opción. Por lo tanto, dentro del bucle suele emplearse la construcción *switch-case* de C para llevar a cabo el procesamiento de las distintas opciones.
2. Un aspecto particular de la función `getopt()` es que establece el valor de distintas variables globales tras invocarse, siendo las más relevantes las siguientes:
    - `char* optarg`: almacena el argumento pasado a la opción actual reconocida, si ésta acepta argumentos. Si la opción no incluye un argumento, entonces `optarg` se establece a `NULL`
    - `int optind`: representa el índice del siguiente elemento en el `argv` (elementos que quedan sin procesar). Se usa frecuentemente para procesar argumentos adicionales del programa que no están asociados a ninguna opción. Un ejemplo de ello es la lista de registros de estudiantes que ha de procesarse en el programa a desarrollar en el siguiente ejercicio.

Responda a las siguientes preguntas:

1. Para representar cada una de las entradas del fichero `/etc/passwd` se emplea el tipo de datos `passwd_entry_t` (estructura definida en `defs.h`). Nótese que muchos de los campos almacenan cadenas de caracteres definidas como arrays de caracteres de longitud máxima prefijada, o mediante el tipo de datos `char*`. La función `parse_passwd()`, definida en `show-passwd.c` es la encargada de inicializar

los distintos campos de la estructura. ¿Cuál es el propósito de la función `clone_string()` que se usa para inicializar algunos de los citados campos tipo cadena? ¿Por qué no es posible en algunos casos simplemente copiar la cadena vía `strcpy()` o realizando una asignación `campo=cadena_existente;`? Justifique la respuesta.

2. La función `strsep()`, utilizada en `parse_passwd()`, modifica la cadena que se desea dividir en tokens. ¿Qué tipo de modificaciones sufre la cadena (variable `line`) tras invocaciones sucesivas de `strsep()`? **Pista**: Consúltese el valor y las direcciones de las variables del programa usando un depurador de C como `gdb`.

Realice las siguientes modificaciones en el programa `show-passwd.c`:

- Añada la opción `-i <inputfile>` para especificar una ruta alternativa para el fichero `passwd`. Hacer una copia de `/etc/passwd` en otra ubicación para verificar el correcto funcionamiento de esta nueva opción.
- Implemente una nueva opción `-c` en el programa, que permita mostrar los campos en cada entrada de `passwd` como valores separados por comas (CSV) en lugar de por `":"`.

## Ejercicio 5