## Makefile (0,5 puntos)

Crea un **Makefile** que permita generar todos los programas del enunciado a la vez y cada uno de ellos por separado. Añade una regla (clean) para borrar todos los binarios y/o ficheros objeto y dejar solo los ficheros fuentes. Los programas deben generarse si, y solo si, ha habido cambios en los ficheros fuentes.

## Control de Errores y Usage (0,5 puntos)

Todos los programas deben incluir un control adecuado de los argumentos usando la función `Usage()` y deben controlar los errores en todas las llamadas al sistema.

## Ejercicio 1: iniciales (2 puntos)

Crea un programa que llamaremos iniciales.c . Este programa recibe un nombre de fichero como parámetro. El fichero contendrá un texto. El programa, creará un nuevo fichero llamado igual que el de entrada, pero añadiéndole "iniciales.txt" al final (si no existe debe crearse y, si existe, queremos que el contenido final incluya solamente el resultado de la última ejecución). El programa debe leer el contenido del fichero de entrada utilizando un buffer de 128 elementos (asume que el fichero de entrada puede tener cualquier tamaño). La salida debe ser la primera letra de cada palabra que contenga el fichero (consideraremos que una palabra es una secuencia de caracteres acabada por uno o más espacios). Por ejemplo, si ejecutamos lo siguiente:

$ echo "hola esto es un fichero de prueba">mi_fichero.txt
$./iniciales mi_fichero.txt
$cat mi_fichero.txt.iniciales.txt
heeufdp

## Ejercicio 2: Cuenta palabras (2.5 puntos)

El programa iniciales_v2 que os damos, ofrece la misma funcionalidad que el programa iniciales pero no genera ningún fichero nuevo, la salida se escribe en la salida estándar. Como sabemos que escribirá una letra por palabra, queremos hacer un programa que cuente palabras utilizando iniciales_v2 y el comando de sistema wc (que con la opción –c cuenta caracteres que le llegan por la entrada estándar).

Haz un programa que llamaremos cuenta_palabras.c que escriba en la salida estándar el número de palabras de un fichero (que recibirá como parámetro). Para ello, debes utilizar los dos programas mencionados (iniciales_v2 y wc) conectándolos mediante una pipe (decide qué tipo de pipe quieres utilizar. Con el fichero del ejemplo anterior, la ejecución sería:

$cuenta_palabras mi_fichero.txt
7

## Ejercicio 3: Selecciona números (2.5 puntos)

Os damos dos ficheros que contienen números en **formato entero**. El primer fichero contiene unos números de los cuales queremos seleccionar solamente unos cuantos. El segundo fichero contiene pares de números que hacen referencia a (posición en primer fichero, cantidad de números a leer).

Se pide hacer un programa que llamaremos selecciona_nums que reciba dos argumentos: el fichero con los números y el fichero con los pares (posición,cantidad), ambos ficheros contendrán números

en formato entero (posición empieza en 0). Tanto el fichero con números como el fichero con los pares contienen solamente números, no hay ningún carácter adicional. Los paréntesis y la "," del ejemplo solo estaban por claridad.

El programa leerá los números seleccionados y los mostrará, en asciii, por la salida std (separa los números con un espacio). Por ejemplo, si nos dicen que el fichero num_0_99 contiene los números del 0 al 99 consecutivos, y el fichero mi_seleccion contiene los pares (0,5)(10,3)(4,1), si ejecutamos:

```
$selecciona_nums num_0_99 mi_seleccion
0 1 2 3 4 10 11 12 4
```

Se pide explícitamente: Comprobar que el fichero con los pares (posición, cantidad) no tiene ningún valor incorrecto, por ejemplo, valores negativos o mayores que el tamaño del fichero con los números. En caso de error, dar un mensaje de error y continuar con el siguiente par. Podeis probar con el fichero mi_seleccion_con_errores para ver si los detectáis.

### Contesta las siguientes preguntas en el fichero respuestas.txt, debes justificar todas (0,5 cada una)

- ¿Que tipo de pipe has decidido utilizar en el ejercicio 2? Escribe en el fichero respuestas.txt la línea de comandos o llamada a sistema q has utilizado para crear la pipe.
- Escribe en el fichero respuestas.txt la línea de comandos para crear un hard-link al fichero nums_0_99 y llámalo hl_nums_0_99. Escribe también el comando para crear un soft-link al fichero hl_nums_0_99 y llámalo sl_nums_0_99.
- Escribe en el fichero respuestas.txt la línea de comandos para obtener el número de inodo y el número de links de los dos ficheros creados en el apartado anterior.
- Escribe en el fichero respuestas.txt la línea de comandos para obtener la lista de sistemas de ficheros que hay instalados en el fichero y cuantos inodos hay en cada uno, así como cuantos hay libres y utilizados. Copia el resultado del comando en respuestas.txt.

### Qué se tiene que hacer

- El Makefile
- Los códigos de los programas en C
- La función Usage() para cada programa
- El fichero respuestas.txt con las respuestas a las preguntas

### Qué se valora

- Que sigas las especificaciones del enunciado
- Que el uso de las llamadas a sistema sea el correcto
- Que se comprueben los errores de **todas** las llamadas al sistema
- Código claro y correctamente indentado
- Que el Makefile tenga bien definidas las dependencias y los objetivos

- La función Usage() que muestre en pantalla cómo debe invocarse correctamente al programa en caso que los argumentos recibidos no sean adecuados.
- El fichero respuestas.txt

### Qué hay que entregar

Un único fichero tar.gz con el código fuente de los programas en C, el Makefile, y las respuestas en respuestas.txt:

```
tar zcvf clab2.tar.gz Makefile respuestas.txt *.c
```