General (1 punto)

- Crea un único Makefile que sirva para generar todos los ejecutables que se piden. El Makefile debe además contener las reglas all y clean.
- Todos los programas que tu hagas deben comprobar el número de parámetros e invocar a la función Usage() en caso de un uso incorrecto.
- Todos los programas deben controlar los errores en todas las llamadas al sistema.

Procesos: (4 puntos)

Crea un programa en C, llámalo **procesos00.c**, que secuencialmente vaya creando tantos procesos como argumentos tenga menos 1 (es decir, el nombre del programa no cuenta). Si se ejecuta sin argumentos, debe mostrar un mensaje de error y acabar. Cada proceso hijo escribe por pantalla el nombre del argumento y acaba. El proceso padre espera a que su hijo acabe, escribe por pantalla el pid del hijo y termina.

Modifica el programa anterior, llámalo **procesos01.c**. Ahora la creación de procesos tiene que ser concurrente. El padre debe esperar a todos sus hijos y escribir el pid de cada hijo por pantalla a medida que vayan acabando.

Modifica el programa anterior. Llámalo **procesos10.c**. El primer argumento de este programa será el nombre de un comando Unix a ejecutar y el resto de argumentos serán las opciones de éste. Haz los cambios necesarios para que todos los hijos modifiquen su imagen (muten) y ejecuten el comando Unix con sus opciones. Es decir, todos los hijos ejecutarán la misma línea de comandos.

Ejemplos:

```
$ procesos10 echo a
a
a
PARE: fill 4676 mort
PARE: fill 4677 mort
$ procesos10 echo a b
a b
a b
a b
PARE: fill 4654 mort
PARE: fill 4653 mort
PARE: fill 4655 mort
```

Signals : (3,25 puntos)

Modifica el programa anterior. Llámalo **signals00.c**. El proceso padre, una vez creados todos sus hijos, debe entrar en una espera activa hasta que todos ellos acaben. Programa el signal pertinente para que cuanto un proceso hijo acabe, el proceso padre escriba por pantalla el resultado de la ejecución del comando Unix (recuerda que en Unix un comando devuelve 0 si ha tenido éxito).

Ejemplo (no existe el usuario 2121):

```
$ ./signals00 grep 2121 /etc/passwd
El proceso 3172 termina con exit code 1
El proceso 3173 termina con exit code 1
El proceso 3174 termina con exit code 1
```

Modifica el programa anterior. Llámalo **signals01.c**. Suprime la espera activa del proceso padre y sustitúyela por una espera bloqueante. Cada segundo, el proceso padre escribirá por pantalla: "procesando".

Ejemplo (el fichero "Hola" no existe bajo el directorio /home, pero el find acaba correctamente):

./signals01 find /home/ -name "Hola"

procesando... procesando... procesando... procesando... procesando... procesando...
procesando... procesando... procesando... procesando...

El proceso 3822 termina con exit code 0

El proceso 3821 termina con exit code 0

El proceso 3820 termina con exit code 0

El proceso 3819 termina con exit code 0

Modifica el programa anterior. Llámalo **signals10.c**. Programa el signal SIGHUP. Cada vez que el proceso padre reciba ese signal, alternará el idioma del mensaje, de "procesando" a "processant" y viceversa. Para probar que funciona…

A. ¿Qué línea de comandos tienes que ejecutar desde el terminal? Escribe la respuesta en un fichero de texto llamado **clab1.txt.**

Se valorará que cada proceso tenga capturados únicamente los signals que va a tratar.

Memoria: (1,75 puntos)

Modifica el programa anterior. Llámalo **mem00.c**. El programa tendrá que reservar dinámicamente un vector que tenga exactamente el mismo número de elementos que argumentos pasados a línea de comando. La reserva se tiene que realizar usando la llamada a librería de C (y no la llamada a sistema). El proceso padre asignará el resultado de la ejecución de los hijos a las correspondientes posiciones en el vector. Cuando acaben todos los hijos, liberará la memoria asignada.

Contesta las siguientes preguntas en el fichero **clab1.txt**

B. ¿Qué fichero consultarías para comprobar el estado del proceso padre y los hijos? Anota el comando realizado y el estado que has comprobado.
C. ¿Qué fichero consultarías para ver las regiones de memoria de un proceso? Anota el comando que utilizarías para verlo y el contenido de las regiones del proceso padre.
D. Relaciona cada una de las variables globales de tu programa con las regiones que has observado, anota el rango de direcciones a las que pertenecen las variables.

A entregar

Debéis entregar un único fichero .tar.gz que contenga el Makefile, todos los ficheros fuentes y el fichero clab1.txt con las respuestas a las preguntas realizadas.