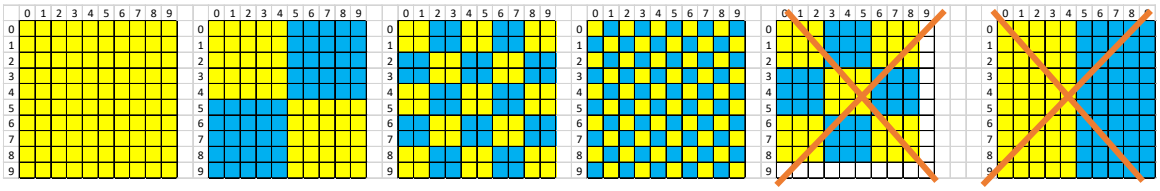
Sistemas Operativos

Examen Práctico
Convocatoria Extraordinaria – Septiembre 2019



Supongamos que tenemos una matriz de datos de 10x10.

Las únicas formas de dividirlas en sub-matrices cuadradas de forma que cada sub-matriz sea procesada por un proceso y no se quede nada sin procesar, son las que no se ven tachadas en la imagen. Las dos ultimas no valen, una porque no se cubre toda la matriz y la otra porque las submatrices no son cuadradas.

Se pretende dividir el trabajo de contar los **ceros**, los **pares no nulos** y los **impares** que hay en la matriz entre varios procesos. Para ello un proceso padre en C rellenará aleatoriamente la matriz con datos enteros entre 0 y 100. Creará el numero de hijos que le indican en un parametro. Cada hijo contará los ceros, pares e impares de su porción de matriz. Esto se repetirá varias veces. Para ello un script en bash llamara varias veces al proceso C.

22	23	57	57	32	83	65	1	47	25
90	52	82	33	66	80	27	36	67	77
90	53	9	53	37	13	8	2	54	30
39	26	43	71	91	17	2	31	36	94
70	80	26	18	19	83	45	59	56	40
74	0	31	100	36	7	94	44	52	38
0	84	12	17	59	57	87	63	24	42
69	54	62	100	92	80	11	13	21	45
68	89	79	98	100	88	9	42	40	38
29	96	41	71	78	2	23	41	92	60

	Ceros	Pares	Impares
Hijo 0	0	12	13
Hijo 1	0	11	14
Hijo 2	2	16	9
Hijo 3	0	14	11
	2	**53**	**47**

En este ejemplo con 4 hijos numerados de 0 a 3, el hijo 2 (inferior-izquierda) cuenta 2 ceros, 16 pares y 9 impares. Los demás cuentan los suyos. Cada hijo le dice al padre sus cuentas y este suma los ceros, los pares y los impares de tdos los hijos, resultando en 2, 53 y 47 respectivamente.

Esto sería una ejecución. El programa en bash llamara varias veces a partec y sumara los totales de ceros, pares e impares de todas las ejecuciones.

Disponeis de los esqueletos de partec y partebash en el directorio examen de vuestro home con comentarios para indicar lo que hay que hacer.
Ademas disponeis de código de ejemplo bash para ayudaros a trabajar con vectores en bash.

Se pide: **Completar sólo las secciones de código marcadas con begin/end** con vuestro código en los esqueletos de partec y partebash para que funcionen correctamente. Podeis añadir funciones nuevas al final, pero no tocar las ya escritas.

Se llamará a partebash con la siguiente sintaxis:

```
partebash <numEjecuciones> <numHijos> <ladoMatriz>
```

donde ***numEjecuciones*** es el numero de veces que se llama a partec, ***numHijos*** es el numero de hijos que se van a crear en partec y ***ladoMatriz*** es el lado de la matriz que se va a crear.

Tras la ejecución se mostrarán el numero totales de ceros, pares e impares que se han contado en todas las ejecuciones por todos los hijos.

El proceso en C: partec (6 puntos)

Completar el proceso en C partec.c partiendo del esqueleto que teneis en vuestro home. El programa recibirá dos parametros, el **numero de hijos** y el **lado** de la matriz, de forma que su tamaño sea **lado x lado**.

Como en función del tamaño de la matriz el número de hijos (bloques cuadrados) será diferente, el programa da un error si se le pasa un numero incorrecto de hijos. Automáticamente muestra un mensaje diciendo los posibles hijos que se pueden usar. Podeis usar ***partecprofesor*** para ejecutarlo, ver su salida.

Si los parametros son correctos no saca nada por pantalla y su codigo de retorno (exit) es 0.
Si los parametros son errorneos saca el siguiente mensaje y su codigo de retorno es 1.

```
$ partecprofesor 16 10
Tamaño de bloque no homogéneo o matrix demasiado grande.
Número de hijos validos para matrices 10x10 : 1 4 25 100
```

Aquí se ha llamado a partec con un numero errorneo de hijos (16) para una matriz de 10x10 (100 elementos) por lo que se queja pero nos dice que se podría llamar con 1, 4, 25 o 100 hijos (como en la imagen). **Para desarrollar utilizar por ejemplo hijos=4 y lado=10 ($partec 4 10)**

- **Se deberá utilizar memoria compartida para acceder a la matriz que genera el padre.**
- **Se utilizarán pipes para que los hijos comuniquen al padre el numero de ceros, pares e impares que han contado en su bloque.**

El padre comprueba que para el tamaño ***ladoMatriz*** que recibe, se puede procesar completa con ***numHijos*** blockes cuadrados. La funcion que comprueba esto ya está desarrollada. Si no son correctos los parametros el padre terminará con exit(1) y si lo son continúa.

El proceso padre crea e inicializa aleatoriamente la matriz de memoria compartida.

Crea un array de estructuras HIJO (definidas en el esqueleto) donde guardará información de los hijos, (pid, pipes, numceros, etc..)

Entra en bucle para crear a los hijos.

> En el bucle el padre comprueba errores y va anotando el pid del hijo en su estructura.
>
> En el bucle el hijo se vincula a la memoria compartida y utiliza una funcion GetBlock (ya desarrollada) que devuelve un array con los datos de su bloque, para que pueda contar ceros, pares e impares. Para contar utilizará funciones que hay que completar. Cuando termina de contar escribe en el pipe que tiene con su padre para enviarle los ceros, pares e impares. Y termina.

Tras el bucle de creación de los hijos el padre espera la terminación de los mismos en un bucle utilizando ***wait*** o ***waitpid***. Cuando detecta que un hijo ha terminado busca su estructura HIJO en el array por el pid y en ella encuentra el pipe en el que leer los datos del hijo (ceros, pares e impares). Lee los datos del pipe y los coloca en la estructura del hijo.

Cuando sale del bucle de terminación, en las estructuras están todos los datos. Los suma y genera los ficheros de salida que leerá el proceso partebash. Estos ficheros tienen la sintaxis **Pnnnn_data.txt** donde **nnnnn** es el **PID** del proceso padre, asi se diferencian los ficheros, puesto que en cada ejecución se crea un padre distinto.

El padre termina con exit(0) si todo es correcto, sin sacar ningun mensaje por pantalla. (Puedes poner los que necesites para debug, pero luego los comentas).

El proceso en BASH: partebash (4 puntos)

El script partebash controlará los parametros ***numEjecuciones***, ***numHijos*** y ***ladoMatriz*** entre los rangos que están fijados en el esqueleto.

Lanzará tantas veces como numEjecuciones indique al proceso partec pasándole numHijos y ladoMartriz.

Comprobará que partec devuevlve 0, lo que indica que su ejecución es correcta. Si es incorrecta terminará mostrando mensaje de error.

Cuando han terminado todas las ejecuciones de partec, en el directorio se encontrará un fichero por cada ejecución con el nombre ***Pnnnnn_data.txt*** y que contiene tres lineas una para los ceros, otra para pares y otra para impares de la matriz completa (la suma de lo que los hijos reportaron). El programa, abrirá uno por uno estos ficheros y leyendo linea por linea sumara los valores a contadores, de forma que al final mostrará estos totales por pantalla.

En el esqueleto se propone trabajar con arrays con la ayuda de funciones ya definidas, pero podéis implementarlo como queráis.

Funciones de ayuda y soporte

- El makefile ya está completo, solo hay que ejecutarlo
- El fichero functions.h esta completo, si necesitas más funciones poner su prototipo aquí.
- El fichero functions.c tiene alguna función que tenéis que completar.
- La mayoría de las funciones ya están desarrolladas.
- Es muy impotante que leais atentamente los comentarios del codigo puesto que os ayudarán bastante
- Hay código srcipt *ejemplo_suma_vectores* que muestra cómo hacer la suma y trabajar con arrays