



## **Sistemas Operativos**

### Práctica

**Lic. Exequiel Aramburu**

[exequiel.aramburu@uader.edu.ar](mailto:exequiel.aramburu@uader.edu.ar)

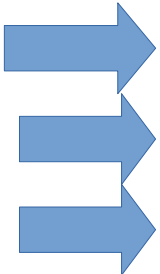


Esta obra está bajo una [Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).

## Agenda

- Hilos en POSIX.
- Hilos. Práctica de Laboratorio utilizando el lenguaje C.
- Condición de carrera. Práctica de Laboratorio utilizando el lenguaje C.
- Administración de tareas GNU/Linux (Crontab).
- Prácticas de administración de tareas programadas en GNU/Linux.
- Actividad extra aúlica. Analizar las administración de tareas programadas de Microsoft Windows.

## Hilos en POSIX



Llamada de hilo	Descripción
Pthread_create	Crea un nuevo hilo
Pthread_exit	Termina el hilo llamador
Pthread_join	Espera a que un hilo específico termine
Pthread_yield	Libera la CPU para dejar que otro hilo se ejecute
Pthread_attr_init	Crea e inicializa la estructura de atributos de un hilo
Pthread_attr_destroy	Elimina la estructura de atributos de un hilo

**Figura 2-14.** Algunas de las llamadas a funciones de Pthreads.

- Para poder utilizar la interfaz de los hilos es necesario incluir la cabecera a `pthread.h`. Además, a la hora de compilar hay que enlazar el código con la opción **-pthread**.

```
gcc hilos.c -o hilos -pthread
```

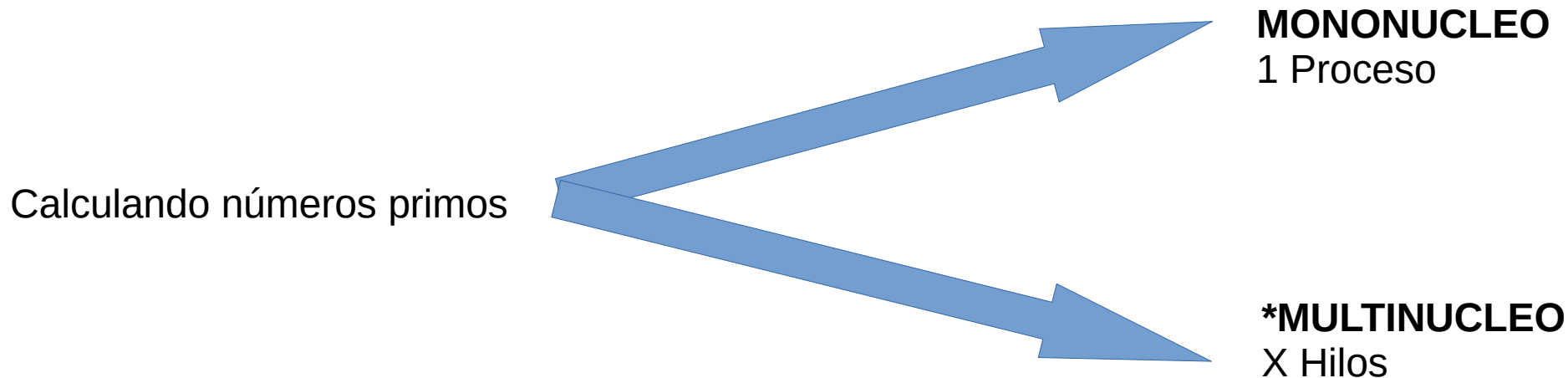
- La creación de un hilo se hace mediante **pthread\_create**. A partir de este punto, si la función no produce error, hay dos hilos de ejecución: el del programa invocante y otro cuyo nombre de función se pasa por parámetro y en nuestro caso se corresponde con **primos\_h**. Dicha función recibe un puntero a datos y devuelve otro, en nuestro caso **&tdata**. Típicamente, el hilo invocante usa el último parámetro de hilo para enviar datos de entrada al nuevo hilo.

```
pthread_create (&h1 , NULL , (void *) primos_h, (void *) &tdata1);
```

- El resultado del hilo se devuelve cuando la función con la que se crea el hilo finaliza. El hilo devuelve un puntero cuyos resultados se pueden recoger, más tarde, con **pthread\_join** desde el hilo padre que lo ha creado.

```
pthread_join (h1 , NULL );
```

## EJEMPLO DE HILOS



## Práctica de hilos

- **DESCARGAR EL CÓDIGO FUENTE**

Del campus -->



Práctica Clase 10 - Código Fuente: Hilos.c

- **COMPILAR EL CÓDIGO FUENTE**

```
gcc hilos.c -o hilos -pthread
```

- **EJECUTAR EL PROGRAMA**

```
./hilos
```

## 2 Terminales

Ejecutando el programa

Ejecutando htop

```
BUSCADOR DE NUMERO PRIMOS - Mi PID es: 29480
Numero Máximo: 100000
Realizar busqueda MonoNucleo[1-Si/2-no]: 1
Hilos [2-12]: 12
Descripción Mononucleo                               Buscando
Mononucleo[0,100000]: 9592                             100000
Tiempo(ms):10776      Tiempo(s):10.776000
Numeros Primos Encontrados[0,100000]: 9592

ESTADO      HILO      RANGO      RESULTADO  PID      Buscando
Terminado   1      [0,8333]      1045      29508     8333
Terminado   2      (8333,16666]  883      29509     16666
Terminado   3      (16666,24999] 834      29510     24999
Terminado   4      (24999,33332] 807      29511     33332
Terminado   5      (33332,41665] 790      29512     41665
Terminado   6      (41665,49998] 773      29513     49998
Terminado   7      (49998,58331] 773      29514     58331
Terminado   8      (58331,66664] 740      29515     66664
Terminado   9      (66664,74997] 748      29516     74997
Terminado  10      (74997,83330] 741      29517     83330
Terminado  11      (83330,91663] 725      29518     91663
Terminado  12      (91663,100000] 733      29519     100000
Tiempo(ms):392      Tiempo(s):0.392000
Numeros Primos Encontrados[0,100000]: 9592

Diferencia(s): 10.384000
Diferencia (X mas rapido): 27

exequiel@Exequiel-PC:~/Escritorio/sistemas operativos/procesos_c$
```

exequiel@Exequiel-PC: ~

Archivo Editar Ver Buscar Terminal Ayuda

```
0[||||| 70.7 ] 3[||||| 97.8 ] 6[||||| 95.6 ] 9[||||| 43.9 ]
1[||||| 82.1 ] 4[||||| 64.5 ] 7[||||| 94.9 ] 10[||||| 80.4 ]
2[||||| 100.0 ] 5[||||| 65.6 ] 8[||||| 99.4 ] 11[||||| 56.4 ]

Mem[|||||] 4.10G/31.2G Tasks: 217, 928 thr, 187 kthr; 10 running
Swp[|||||] 0K/16.8K Load average: 1.66 0.81 0.58
Uptime: 01:45:14
```

Main	I/O										
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
29687	exequiel	20	0	162M	1712	1596	S	852	0.0	1:33.43	./hilos
29711	exequiel	20	0	162M	1712	1596	R	101	0.0	0:08.96	./hilos
29716	exequiel	20	0	162M	1712	1596	R	101	0.0	0:09.22	./hilos
29712	exequiel	20	0	162M	1712	1596	R	100	0.0	0:09.10	./hilos
29714	exequiel	20	0	162M	1712	1596	R	100	0.0	0:09.16	./hilos
29715	exequiel	20	0	162M	1712	1596	R	100	0.0	0:09.28	./hilos
29717	exequiel	20	0	162M	1712	1596	R	100	0.0	0:09.34	./hilos
29718	exequiel	20	0	162M	1712	1596	R	100	0.0	0:09.32	./hilos
29713	exequiel	20	0	162M	1712	1596	R	99.5	0.0	0:09.15	./hilos
1969	exequiel	20	0	4212M	331M	114M	S	54.2	1.0	12:02.54	cinnamon --replace
3978	exequiel	20	0	604M	44552	3328	S	17.9	0.1	0:15.05	/usr/libexec/gnome-terminal-server
29677	exequiel	20	0	15560	6332	3568	R	4.5	0.0	0:00.91	htop

## Práctica

Ejecute el programa hilos y registre los resultados en la siguiente tabla:

Procesador: X



cat /proc/cpuinfo

Núcleos/hilos: X/X

Hilos/Buscar	Primos encontrados	1	2	4	6	8	10	12
200.000	X	X segundos						
500.000								
1.000.000								



## Práctica

Resultado en mi PC.

Procesador: Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz

Núcleos/hilos: 6 núcleos / 12 hilos

Hilos/Buscar	Primos encontrados	1	2	4	6	8	10	12
200.000	17.984	4,36	3,17	1,972	1,45	1,26	1,12	1,12
500.000	41.538	23,99	17,79	10,62	7,54	6,32	5,53	5,32
1.000.000	78.498	90,31	66,87	39,25	27,57	23,03	20,15	18,63

# Práctica de condición de carrera

## CONDICIÓN DE CARRERA

Una condición de carrera sucede cuando múltiples procesos o hilos leen y escriben datos de manera que el resultado final depende del orden de ejecución de las instrucciones en los múltiples procesos. Consideremos dos casos sencillos.

Como primer ejemplo, suponga que dos procesos, P1 y P2, comparten la variable global *a*. En algún punto de su ejecución, P1 actualiza *a* al valor 1 y, en el mismo punto en su ejecución, P2 actualiza *a* al valor 2. Así, las dos tareas compiten en una carrera por escribir la variable *a*. En este ejemplo el «perdedor» de la carrera (el proceso que actualiza el último) determina el valor de *a*.

*Stallings, W., Aguilar, L. J., Doderio, J. M., Torres, E., & Mora, M. K. (1997). Sistemas operativos (Vol. 732). Prentice Hall.*

**Depurar** programas que contienen **condiciones de carrera no es nada divertido**. Los resultados de la mayoría de las ejecuciones de prueba están bien, pero en algún momento poco frecuente ocurrirá algo extraño e inexplicable.

*Tanenbaum, A., Stallings, W., & Kerrish, M. Sistemas Operativos modernos., 3ra edición.*

## Práctica de condición de carrera

- DESCARGAR EL CÓDIGO FUENTE

Del campus -->



Práctica Clase 10 - Código Fuente: Condición\_de\_Carrera.c

- COMPILAR EL CÓDIGO FUENTE

**gcc carrera\_condicion.c -o carrera\_condicion -pthread**

- EJECUTAR EL PROGRAMA

**./carrera\_condicion**



# CRONTAB

Linux Administration

# Configuración del Crontab

🎯 A crontab file has five fields for specifying:

```
* * * * * command to be executed
- - - - -
| | | | |
| | | | +----- **DAY OF WEEK** (0-6) (Sunday=0)
| | | +----- **MONTH** (1-12)
| | +----- **DAY OF MONTH** (1-31)
| +----- **HOUR** (0-23)
+--- **MINUTE** (0-59)
```

# Configuración del Crontab

También puede incluir ciertos caracteres especiales en el componente de programación de una expresión de Cron para facilitar la programación:

- `*`: en las expresiones de Cron, el asterisco es la variable comodín que representa “todo”. Por lo tanto, una tarea programada con `* * * * *` se ejecutará cada minuto de cada hora de cada día de cada mes.
- `,`: las comas separan los valores de programación para crear una lista. Si desea que una tarea se ejecute al comienzo y a la mitad de cada hora, en lugar de escribir dos tareas separadas (por ejemplo, `0 * * * * *` y `30 * * * * *`), podría lograr la misma funcionalidad con una sola (`0,30 * * * * *`).
- `-`: el guión representa una variedad de valores en el campo de programación. En lugar de tener 30 tareas programadas por separado para un comando que desee ejecutar durante los primeros 30 minutos de cada hora (como en el caso de `0 * * * * *`, `1 * * * * *` y `2 * * * * *`, entre otros), podría programarlo con el valor `0-29 * * * * *`.
- `/`: puede usar una barra diagonal con un asterisco para expresar un valor de paso. Por ejemplo, en lugar de escribir ocho tareas de Cron separadas para ejecutar un comando cada tres horas (como en el caso de `0 0 * * * *`, `0 3 * * * *` y `0 6 * * * *`, entre otros), podría programarlo con el siguiente valor: `0 */3 * * * *`.

## Comandos crontab

### Listado de trabajos Cron existentes

Podemos enumerar todos los trabajos de Cron sin abrir el archivo de configuración crontab usando el siguiente comando

```
crontab -l
```

### Agregar / modificar entradas de Crontab

Para editar la entrada crontab, podemos usar `-e` opción como se muestra a continuación.

```
crontab -e
```

## Ejecute múltiples tareas usando un solo cron

```
* * * * * /home/scripts/backup.sh; /home/scripts/scriptp.sh
```

Se pueden separar varias tareas / trabajos usando un punto y coma (;) y se puede asignar a una sola expresión cron.



## EJEMPLOS

## DETALLE

**\* \* \* \* \*** /home/aramburu/scripts/script.sh

→ cada minuto.

**0 \* \* \* \*** /home/aramburu/scripts/script.sh

→ cada hora, en el minuto 0.

**0 4 \* \* \*** /home/aramburu/scripts/script.sh

→ diaria a las 4 de la mañana.

**0 9,20 \* \* \*** /home/aramburu/scripts/script.sh

→ A las 9 de la mañana y a las 20 de la tarde (2 veces al día)

**0 9-20 \* \* \*** /home/aramburu/scripts/script.sh

→ cada hora desde las 9 de la mañana hasta las 20 de la tarde.

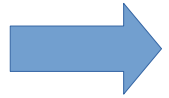
**0 9-20 \* \* 1-5** /home/aramburu/scripts/script.sh

→ cada hora entre las 9 de la mañana y las 20 de la tarde de lunes a viernes.

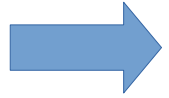
**\*/10 \* \* \* \*** /home/aramburu/scripts/script.sh

→ cada 10 minutos.

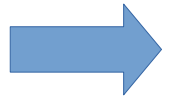
## ACTIVIDAD DE LABORATORIO



Cree un directorio llamado **Clase10** en su home.



Programe un cron para que se ejecute cada minuto y borre el archivo llamado prueba.txt si existe en el directorio antes creado.



Cree un archivo llamado prueba.txt y verifique el correcto funcionamiento del cron.

## Actividad extra aúlica

# Analizar la administración de tareas programadas en Microsoft Windows

