

# MiniTP Shell y Procesos 2023

26 MARZO

## SOR – Sistemas Operativos y Redes (Com1)

Profesor/es: Lic. Mariano Vargas - TUI. Augusto Liali

Alumno: Guillermo Dominguez

```
static void StartElement(void *voidContext,
                        const xmlChar *name,
                        const xmlChar **attributes)
{
    Context *context = (Context *)voidContext;

    if (COMPARE((char *)name, "TITLE"))
    {
        context->title = "";
        context->addTitle = true;
    }

    (void) attributes;
}

// libxml end element callback function
static void EndElement(void *voidContext,
                        const xmlChar *name)
{
    Context *context = (Context *)voidContext;

    if (COMPARE((char *)name, "TITLE"))
        context->addTitle = false;
}

// Text handling helper function
static void handleCharacters(Context *context,
                            const xmlChar *chars,
                            int length)
{
    if (context->addTitle)
        context->title.append((char *)chars, length);
}

// libxml PCDATA callback function
static void Characters(void *voidContext,
                      const xmlChar *chars,
                      int length)
{
    Context *context = (Context *)voidContext;

    handleCharacters(context, chars, length);
}

static void cdata(void *voidContext,
                  const xmlChar *chars,
                  int length)
{
    Context *context = (Context *)voidContext;

    handleCharacters(context, chars, length);
}
```



Universidad de  
General Sarmiento

---

# Sistemas Operativos y Redes

## MiniTP – Shell y Procesos

### Condiciones para Aprobar

Para la evaluación del presente trabajo, deben realizar los siguientes puntos:

- El trabajo es individual
- Enviar el trabajo en formato digital dentro de los plazos establecidos. Adjuntar el código fuente de su implementación y un informe del trabajo realizado punto por punto, dificultades encontradas y soluciones propuestas.

### Puntaje / Calificación:

El presente trabajo se califica con las notas:

- I (insuficiente)
- A- (aprobado menos, no puede tener dos A- en la cursada),
- A (aprobado)
- A+ (aprobado más, redondea para arriba la nota final en caso de promocionar)

### Recuperatorio:

En caso de no aprobar tiene un plazo de una semana para entregar el TP con las correcciones indicadas más ejercicios adicionales que se agregaran al enunciado. En recuperatorio no se pone A +.

### Los objetivos de este trabajo son:

- Familiarizar al alumno con la línea de comandos.
- Familiarizar al alumno con las nociones de Proceso y Thread.

## Ejercicio 1 – Shell y Terminal

Realice un script de shell tal que realice las siguientes tareas:

- Debe recibir por parámetro una palabra y debe crear un directorio con dicho nombre en tu home de usuario.
- Dentro de ese directorio debe crear un archivo .txt
- Debe agregar al archivo anterior el listado de todos los archivos de la computadora que terminan con la extensión .txt, y además de los nombres de los archivos se tienen que ver los permisos de los mismos.
- Al final del archivo y del listado debe agregar la fecha y la hora del sistema
- Cuando el script termine debe mostrar por pantalla el contenido del archivo.

Puede usar los siguientes operadores y comandos:

- pipe | , redirección > , redirección concatenando >>, grep, ls y cat

### Resolucion:

```
Executable File | 30 lines (27 sloc) | 672 Bytes
1  #!/bin/bash
2
3  function Error(){
4      echo "Error, se debe ingresar un parametro"
5  }
6
7  #Verifico que el usuario haya ingresado un solo parametro
8  if [ $# -eq 1 ]
9  then
10     #Defino las variables a utilizar
11     now=$(date +%d/%m/%Y)
12     nameDir=$1
13     txtFiles=$(find ~ -type f -iname "*.txt")
14     #Me 'muevo' al directorio home del usuario
15     cd $HOME
16     #Verifico que si existe el directorio, en el caso que si accedo, caso contrario lo genero
17     if [ -d "$nameDir" ]
18     then
19         cd ${nameDir}
20     else
21         mkdir ${nameDir}
22         cd ${nameDir}
23     fi
24     echo -e "ARCHIVOS CON EXTENSION .TXT \n${txtFiles}\n" > report.txt
25     echo "Reporte generado: ${now}" >> report.txt
26 else
27     Error
28 fi
29 exit
30
```

En esta implementacion se cumplen con los puntos citados en la consigna y adicionalmente se aplican dos verificaciones adicionales. En primer lugar se controla que el usuario ingrese un solo parametro y en segundo lugar se verifica que el directorio no exista.

```
guillermo@jgdesk: ~/carpetaEj1

guillermo@jgdesk:~/Documentos/SOR_Practicas/02 - Shell y Procesos (TP)$ ./ej1.sh carpetaEj1
guillermo@jgdesk:~/Documentos/SOR_Practicas/02 - Shell y Procesos (TP)$ cd
guillermo@jgdesk:~$ ls -l
total 44
drwxrwxr-x 2 guillermo guillermo 4096 mar 26 23:13 carpetaEj1
drwxr-xr-x 4 guillermo guillermo 4096 mar 26 11:26 Descargas
drwxr-xr-x 6 guillermo guillermo 4096 mar 26 18:49 Documentos
drwxrwxr-x 3 guillermo guillermo 4096 mar 19 19:46 eclipse
drwxr-xr-x 2 guillermo guillermo 4096 mar 19 06:29 Escritorio
drwxr-xr-x 3 guillermo guillermo 4096 mar 12 18:32 Imágenes
drwxr-xr-x 2 guillermo guillermo 4096 mar 12 14:40 Música
drwxr-xr-x 2 guillermo guillermo 4096 mar 12 14:40 Plantillas
drwxr-xr-x 2 guillermo guillermo 4096 mar 12 14:40 Público
drwx----- 12 guillermo guillermo 4096 mar 13 22:54 snap
drwxr-xr-x 3 guillermo guillermo 4096 mar 20 16:06 Videos
guillermo@jgdesk:~$ cd carpetaEj1/
guillermo@jgdesk:~/carpetaEj1$ ls -l
total 8
-rw-rw-r-- 1 guillermo guillermo 7755 mar 26 23:13 report.txt
guillermo@jgdesk:~/carpetaEj1$ nano report.txt
guillermo@jgdesk:~/carpetaEj1$
```

Se evidencia la creacion del directorio (carpetaEj1) y generacion de archivo con la informacion solicitada en el enunciado (report.txt.).

```
GNU nano 6.2 report.txt
ARCHIVOS CON EXTENSION .TXT
/home/guillermo/.eclipse/org.eclipse.oomph.setup/installer/installRoot.txt
/home/guillermo/.eclipse/org.eclipse.oomph.setup/setups/brandingNotificationURIs.txt
/home/guillermo/.eclipse/org.eclipse.oomph.jreinfo/infos.txt
/home/guillermo/.pki/nssdb/pkcs11.txt
/home/guillermo/.vscode/extensions/ms-ceintl.vscode-language-pack-es-1.76.2023030809/ThirdPartyNo
/home/guillermo/.vscode/extensions/tabbline.tabnine-vscode-3.6.42/LICENSE.txt
/home/guillermo/Descargas/eclipse-installer/features/org.eclipse.ecf.filetransfer.httpclient5.fea
/home/guillermo/Descargas/eclipse-installer/features/org.eclipse.ecf.filetransfer.feature_3.14.18
/home/guillermo/Descargas/eclipse-installer/plugins/org.eclipse.justj.openjdk.hotspot.jre.minimal
/home/guillermo/snap/discord/158/.pki/nssdb/pkcs11.txt
/home/guillermo/snap/firefox/common/.mozilla/firefox/5118x2sz.default/ServiceWorkerCacheStorage
/home/guillermo/snap/firefox/common/.mozilla/firefox/5118x2sz.default/pkcs11.txt
/home/guillermo/snap/firefox/common/.mozilla/firefox/5118x2sz.default/AlternateServices.txt
/home/guillermo/snap/chronium/2392/.pki/nssdb/pkcs11.txt
/home/guillermo/snap/chronium/2381/.pki/nssdb/pkcs11.txt
/home/guillermo/snap/chronium/common/chronium/Subresource Filter/Unindexed Rules/9.43.0/LICENSE.t
/home/guillermo/snap/chronium/common/chronium/Profile 1/Extensions/e1nadpbcfbfnnbkopoojfejhkhdbie
/home/guillermo/snap/chronium/common/chronium/Profile 1/Extensions/glgghmpiocklfepjocnangkkbiglid
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/f278ba7242a4b
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/6bf6ab7f94a21
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/2eb603460498a
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/314c0a43f2ada
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/579544fd7d044
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/1ba008ec2cec5
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/c7d67cba325cc
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/212a1fef72049
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/32cadb2b6d359
/home/guillermo/snap/chronium/common/chronium/Profile 1/Service Worker/CacheStorage/379f1cbab5b08
/home/guillermo/snap/chronium/common/chronium/Default/Extensions/e1nadpbcfbfnnbkopoojfejhkhdbieeh
/home/guillermo/snap/chronium/common/chronium/Default/Extensions/glgghmpiocklfepjocnangkkbiglidon
/home/guillermo/snap/chronium/common/chronium/Default/Service Worker/CacheStorage/1e538fbb9a41157
/home/guillermo/snap/chronium/common/chronium/Default/Service Worker/CacheStorage/379f1cbab5b08b6
/home/guillermo/snap/chronium/common/chronium/ZxcvbnData/3/female_names.txt
/home/guillermo/snap/chronium/common/chronium/ZxcvbnData/3/surnames.txt
/home/guillermo/snap/chronium/common/chronium/ZxcvbnData/3/us_tv_and_film.txt
/home/guillermo/snap/chronium/common/chronium/ZxcvbnData/3/english_wikipedia.txt
/home/guillermo/snap/chronium/common/chronium/ZxcvbnData/3/passwords.txt
/home/guillermo/snap/chronium/common/chronium/ZxcvbnData/3/male_names.txt
/home/guillermo/snap/telegram-desktop/4654/.local/share/TelegramDesktop/log.txt
/home/guillermo/snap/telegram-desktop/4654/.local/share/TelegramDesktop/log_start0.txt
/home/guillermo/snap/telegram-desktop/4654/.local/share/TelegramDesktop/log_start1.txt
/home/guillermo/eclipse-workspace/.metadata/.plugins/org.eclipse.pde.core/SavedExternalPluginList
/home/guillermo/.cache/tracker3/files/locale-for-miner-apps.txt
/home/guillermo/.cache/tracker3/files/last-crawl.txt
/home/guillermo/.cache/tracker3/files/first-index.txt

Reporte generado: 19/03/2023
```

Reporte final, en el mismo se evidencia que liste los archivos .txt en el sistema e imprime al final del archivo la fecha en la cual se genero el informe.

Quedo pendiente de este ejercicio, listar los permisos de los archivos .txt.



## Ejercicio 2 – Estados de un Proceso

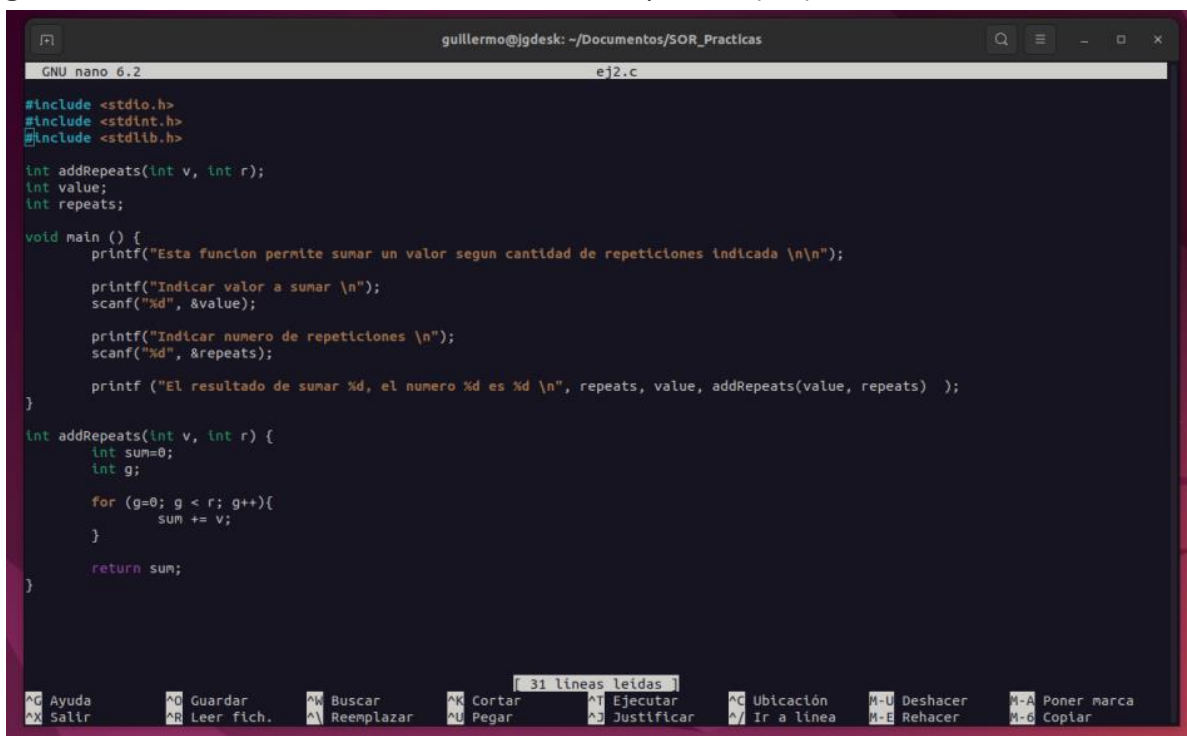
En esta parte vamos a aplicar nuestros conocimientos de procesos y sus estados.

- Realizar un programa en C compuesto de instrucciones que realizan cálculos (operaciones aritméticas) y operaciones de I/O (leer un input del usuario). Compilar, ejecutar su programa y visualizar los estados por los que pasa. Puede usar la herramienta htop.
- Ejecutar su programa y demostrar mediante capturas de pantalla del programa htop que su programa efectivamente cambia de estados.

### Resolucion:

Se genero un programa en C que solicita al usuario dos valores, con los cuales realiza la suma del primer valor de forma consecutiva segun la cantidad de repeticiones ingresadas como segundo parametro.

El programa recién mencionado cuenta con dos interrupciones (I/O).



```
GNU nano 6.2 ej2.c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int addRepeats(int v, int r);
int value;
int repeats;

void main () {
    printf("Esta funcion permite sumar un valor segun cantidad de repeticiones indicada \n\n");
    printf("Indicar valor a sumar \n");
    scanf("%d", &value);

    printf("Indicar numero de repeticiones \n");
    scanf("%d", &repeats);

    printf ("El resultado de sumar %d, el numero %d es %d \n", repeats, value, addRepeats(value, repeats) );
}

int addRepeats(int v, int r) {
    int sum=0;
    int g;

    for (g=0; g < r; g++){
        sum += v;
    }

    return sum;
}
```

A continuacion se puede apreciar el paso a paso durante la ejecucion del programa (terminal derecha) y el seguimiento de estados del programa *ej2* (resaltado color naranja) mediante el programa *htop* (terminal izquierda).

Previo a la ejecucion del programa ej2, nos centramos en el proceso bash (PID 9297).

Al ejecutar el programa ej2, solicita el ingreso de valores por parte del usuario con lo cual el proceso se encuentra en estado S (Sleeping). Es imperceptible el cambio de estado R (Running) a S, al momento de lanzar el programa.

Al finalizar la ejecucion del programa ej2, se elimina el subprocesso de Bash.

En resumen, es imperceptible el cambio de estado *R* (Running) a *S* (Sleeping) al momento de lanzar el programa, pero si se observa claramente que al momento de solicitar un input por parte del usuario se cambia al estado *S*.

## Ejercicio 3 – Procesos y Fork

Fork es la llamada al sistema que permite crear nuevos procesos.

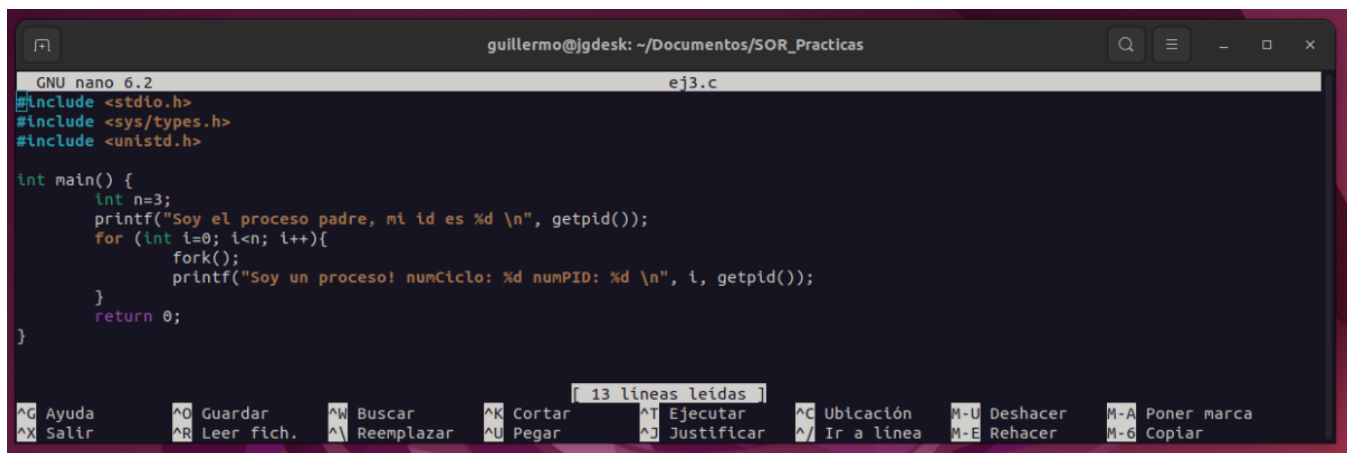
- Compilar, ejecutar el siguiente programa. Describir y justificar el output. Dibujar el árbol de proceso padre-hijo que se van generando para  $n=3$ .

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
    int n=3;
    for (i=0; i<n; i++){
        fork();
        printf("Soy un proceso!\n");
    }
    return 0;
}
```

### Resolucion:

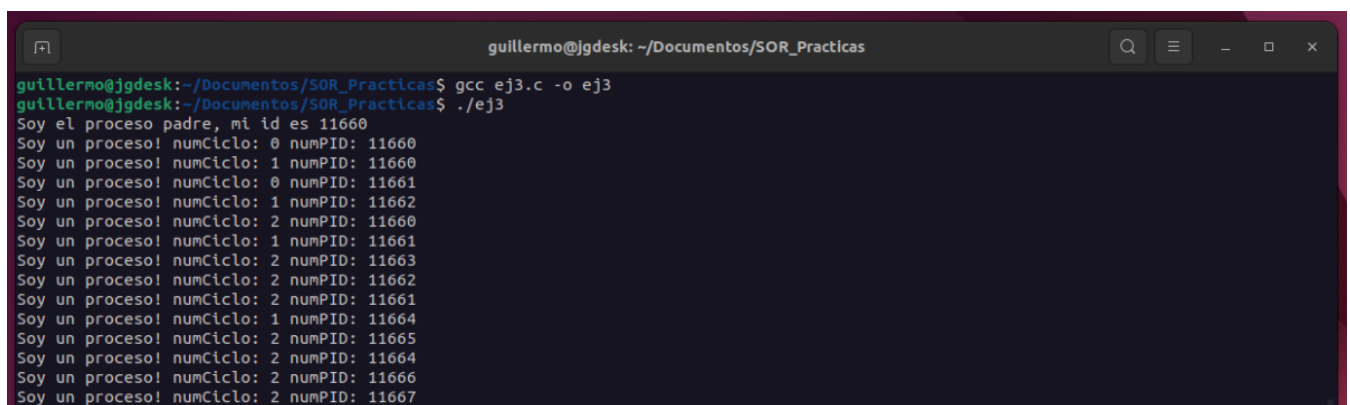
Al momento de ejecutar el programa imprimio en pantalla 12 veces “Soy un proceso!”, me fue imposible detectar en que proceso se originaba el mensaje. Teniendo presente el problema recién mencionado, modifique el mensaje para que me indique el ID del proceso padre, numero de ciclo y el proceso del cual se originaba el mensaje, se puede evidencia esto en las siguientes imágenes.



```
guillermo@jgdesk: ~/Documentos/SOR_Practicas
GNU nano 6.2 ej3.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

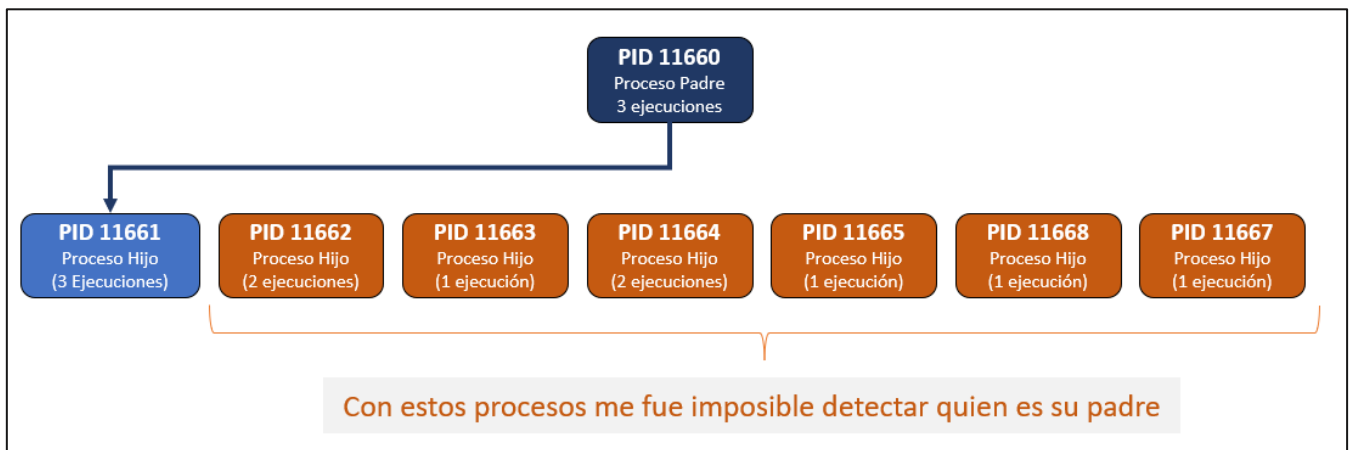
int main() {
    int n=3;
    printf("Soy el proceso padre, mi id es %d \n", getpid());
    for (int i=0; i<n; i++){
        fork();
        printf("Soy un proceso! numCiclo: %d numPID: %d \n", i, getpid());
    }
    return 0;
}
```

Programa con las modificaciones para identificar el proceso *padre* y el proceso que origina el mensaje.



```
guillermo@jgdesk:~/Documentos/SOR_Practicas$ gcc ej3.c -o ej3
guillermo@jgdesk:~/Documentos/SOR_Practicas$ ./ej3
Soy el proceso padre, mi id es 11660
Soy un proceso! numCiclo: 0 numPID: 11660
Soy un proceso! numCiclo: 1 numPID: 11660
Soy un proceso! numCiclo: 0 numPID: 11661
Soy un proceso! numCiclo: 1 numPID: 11662
Soy un proceso! numCiclo: 2 numPID: 11660
Soy un proceso! numCiclo: 1 numPID: 11661
Soy un proceso! numCiclo: 2 numPID: 11663
Soy un proceso! numCiclo: 2 numPID: 11662
Soy un proceso! numCiclo: 2 numPID: 11661
Soy un proceso! numCiclo: 1 numPID: 11664
Soy un proceso! numCiclo: 2 numPID: 11665
Soy un proceso! numCiclo: 2 numPID: 11664
Soy un proceso! numCiclo: 2 numPID: 11666
Soy un proceso! numCiclo: 2 numPID: 11667
```

Salida en pantalla



En este ejercicio pude evidenciar varios conceptos dictados durante la cursada:

- El cambio de proceso del scheduler.
- La importancia de aplicar técnicas para mantener un “orden” en la ejecución de los procesos.
- Los procesos conservan su estado al momento de ser retirados del CPU, esto se evidencia con el numero de ciclo (0, 1, 0 ,1 ,2 ,1 ,etc.).



## Ejercicio 4 – Threads

El siguiente programa ejecuta la función `do_nothing()` cinco veces. Esta función sólo espera 2 segundos y continúa:

```
#include <stdio.h> //incluimos la libreria de estandar input/output
#include <unistd.h> //para hacer sleep
#include <time.h> //para inicializar el tiempo

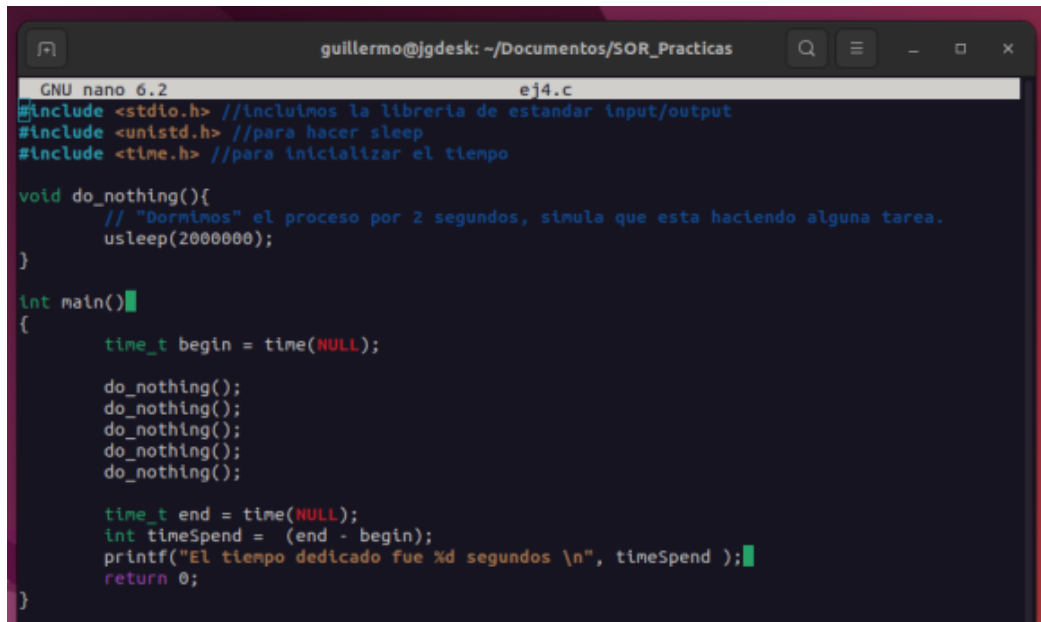
void do_nothing(int microseconds){
    usleep(2000000); //esperar 2 segundos, 1 millon de microsegundos en 1 segundo
                    //dormir el proceso, simula que esta haciendo alguna tarea
}

int main() {
    do_nothing();
    do_nothing();
    do_nothing();
    do_nothing();
    do_nothing();
    return 0;
}
```

- Con la función `time`, medir el tiempo que tarda el programa anterior.
- Modificar el programa anterior para que cada una de las 5 llamadas a la función `do_nothing()` se ejecute por un thread.
- Medir el tiempo que tarda su nuevo programa. Qué diferencias observa? Porque?

### Resolucion:

En primer lugar, se implemento la funcion `time` para medir el tiempo dedicado al ejecutar el programa.



```
guillermo@jgdesk: ~/Documentos/SOR_Practicas
GNU nano 6.2 ej4.c
#include <stdio.h> //incluimos la libreria de estandar input/output
#include <unistd.h> //para hacer sleep
#include <time.h> //para inicializar el tiempo

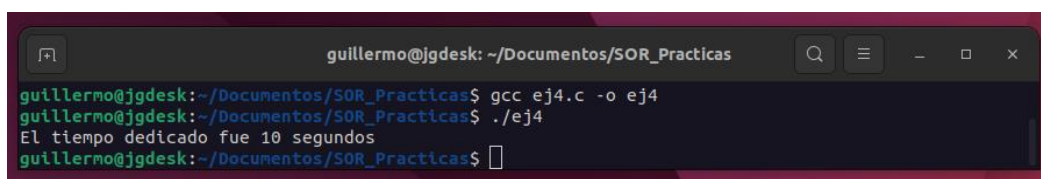
void do_nothing(){
    // "Dormimos" el proceso por 2 segundos, simula que esta haciendo alguna tarea.
    usleep(2000000);
}

int main(){
{
    time_t begin = time(NULL);

    do_nothing();
    do_nothing();
    do_nothing();
    do_nothing();
    do_nothing();

    time_t end = time(NULL);
    int timeSpend = (end - begin);
    printf("El tiempo dedicado fue %d segundos \n", timeSpend );
    return 0;
}
```

Implementacion de funcion `time` e impresión en pantalla de los segundos dedicados en la ejecucion.



```
guillermo@jgdesk:~/Documentos/SOR_Practicas$ gcc ej4.c -o ej4
guillermo@jgdesk:~/Documentos/SOR_Practicas$ ./ej4
El tiempo dedicado fue 10 segundos
guillermo@jgdesk:~/Documentos/SOR_Practicas$
```

Resultado de la ejecucion del programa ej4, con esta implementacion se dedicaron **10 segundos** para ejecutar el programa completo.

```
guillermo@jgdesk: ~/Documentos/SOR_Practicas
GNU nano 6.2 ej4.c
#include <stdio.h> // Libreria de estandar input/output
#include <unistd.h> // Libreria sleep
#include <time.h> // Libreria tiempo
#include <pthread.h> // Libreria threads

void *do_nothing(void *arg){
    // "Dormimos" el proceso por 2 segundos, simula que esta haciendo alguna tarea.
    usleep(2000000);
}

int main()
{
    time_t begin = time(NULL);

    pthread_t t1;
    pthread_t t2;
    pthread_t t3;
    pthread_t t4;
    pthread_t t5;

    pthread_create(&t1, NULL, &do_nothing, NULL);
    pthread_create(&t2, NULL, &do_nothing, NULL);
    pthread_create(&t3, NULL, &do_nothing, NULL);
    pthread_create(&t4, NULL, &do_nothing, NULL);
    pthread_create(&t5, NULL, &do_nothing, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);
    pthread_join(t4, NULL);
    pthread_join(t5, NULL);

    time_t end = time(NULL);
    int timeSpend = (end - begin);
    printf("El tiempo dedicado fue %d segundos \n", timeSpend );
    return 0;
}
```

Implementacion de *threads* para ejecutar la funcion *do\_nothing()*.

```
guillermo@jgdesk: ~/Documentos/SOR_Practicas
guillermo@jgdesk:~/Documentos/SOR_Practicas$ gcc ej4.c -o ej4 -pthread
guillermo@jgdesk:~/Documentos/SOR_Practicas$ ./ej4
El tiempo dedicado fue 2 segundos
guillermo@jgdesk:~/Documentos/SOR_Practicas$
```

Como resultado de la ejecucion del programa *ej4* con la implementacion de *threads*, se logro ejecutar por completo en **2 segundos**.

En este ejercicio se logra observar claramente la genialidad de “trabajar” con hilos!! Se logro ejecutar el programa reduciendo el tiempo al minimo (2 segundos) dado que cada hilo adelantaba las instrucciones de dicho proceso en forma paralela y posteriormente el CPU solo dedicaba rafagas para ejecutar el resultado de cada hilo, con todo “pre-cocinado”.