



Nombre: Josué Merino

NRC: 2537

Examen Práctico 2

Se crea un archivo .c para introducir el código, mediante el siguiente comando:

```
jos@jos:~$ nano caso_2.c
```

Se crea el archivo .c con el nombre **caso_2.c**, se digita el siguiente código y se lo guarda:

```
GNU nano 7.2 caso_2.c *
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

int main(){
    size_t size = 400 * 1024 * 1024;
    void *ptr = malloc(size);
    if(!ptr){
        printf("Error: No se pudo asignar memoria \n");
        return 1;
    }
    printf("400 MB asignados. El programa esta generando carga constante.\n");
    while(1){
        sleep(10);
    }
    free(ptr);
    return 0;
}
```

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea

Se confirma que el archivo se creó:



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

```
jos@jos:~$ nano caso_2.c
jos@jos:~$ ls
caso_2.c  Documentos  Imágenes  Plantillas  proceso1  proceso.c  Público  Videos
Descargas Escritorio  Música    proceso     proceso1.c proyecto  snap
jos@jos:~$
```

Se procede a la ejecución del archivo:

```
jos@jos:~$ gcc caso_2.c -o caso_2
jos@jos:~$
```

```
jos@jos:~$ gcc caso_2.c -o caso_2
jos@jos:~$ ls
caso_2      Escritorio  proceso     proyecto
caso_2.c    Imágenes   proceso1     Público
Descargas   Música      proceso1.c  snap
Documentos  Plantillas proceso.c    Videos
jos@jos:~$
```



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Para el siguiente paso el entorno de interfaces de líneas de comandos que se manejará es el siguiente:

```
jos@jos: ~  
jos@jos:~$ http  
jos@jos:~$ ./caso_2  
jos@jos:~$ ps  
jos@jos:~$ pgrep caso_2
```

Se procede a ejecutar con: `./caso_2`:

```
jos@jos:~$ ./caso_2  
400 MB asignados. El programa esta generando carga constante.  
jos@jos:~$ pgrep caso_2  
4126  
jos@jos:~$
```

Se verifica el proceso utilizando el comando `ps -a`:



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

```
josh@josh: ~  
josh@josh:~$ ps -a  
  PID TTY          TIME CMD  
 1762 pts/2        00:00:00 gnome-session-b  
 4126 pts/1        00:00:00 caso_2  
 4141 pts/2        00:00:00 ps  
josh@josh:~$
```

Mediante el htop visualizamos los recursos que consume el proceso caso_2:

```
josh@josh: ~  
0[|||||] 7.3% Tasks: 111, 347 thr, 95 kthr; 1 running  
1[|||] 1.3% Load average: 0.03 0.14 0.11  
2[|] 0.0% Uptime: 00:23:19  
Mem[|||||] 854M/8.67G  
Swp[|] 0K/4.00G  
Main 1/0  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
2847 josh 20 0 704M 68200 50852 S 2.0 0.7 0:11.77 /usr/libexec/gnome-terminal-server  
2848 josh 20 0 704M 68200 50852 S 0.0 0.7 0:00.00 /usr/libexec/gnome-terminal-server  
2849 josh 20 0 704M 68200 50852 S 0.0 0.7 0:00.00 /usr/libexec/gnome-terminal-server  
2851 josh 20 0 704M 68200 50852 S 0.0 0.7 0:00.00 /usr/libexec/gnome-terminal-server  
2852 josh 20 0 704M 68200 50852 S 0.0 0.7 0:00.02 /usr/libexec/gnome-terminal-server  
2853 josh 20 0 704M 68200 50852 S 0.0 0.7 0:00.01 /usr/libexec/gnome-terminal-server  
2854 josh 20 0 19824 5248 3840 S 0.0 0.1 0:00.07 bash  
2922 root 20 0 1434M 34288 21760 S 0.0 0.4 0:01.78 /usr/lib/snapd/snapd  
3310 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.71 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3312 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3313 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3314 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.07 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3315 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3316 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3317 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3327 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3340 josh 20 0 2815M 62700 47984 S 0.0 0.7 0:00.00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.  
3998 josh 20 0 704M 68200 50852 S 0.0 0.7 0:00.00 /usr/libexec/gnome-terminal-server  
4009 josh 20 0 19692 4992 3712 S 0.0 0.1 0:00.02 bash  
4059 josh 20 0 19692 5248 3840 S 0.0 0.1 0:00.01 bash  
4064 josh 20 0 19692 5120 3840 S 0.0 0.1 0:00.02 bash  
4126 josh 20 0 402M 1280 1280 S 0.0 0.0 0:00.01 ./caso_2  
F1 help F2 setup F3 search F4 filter F5 free F6 sort by F7 nice F8 nice F9 kill F10 quit
```

Explicación y Resumen

El caso de estudio #2 sugiere una visualización de un proceso que asigna 400 MB de memoria y ejecuta un bucle infinito para simular una carga mediante el comando **htop** constante en el sistema.

Para empezar se realiza la creación del archivo en **lenguaje C** para posteriormente ejecutar y visualizar el proceso. La creación se la realizó con el comando: **nano caso_2.c** . Dentro de este archivo se procede a digitar el código que utiliza librerías como: **stdlib**, **unistd** y **stdio**.

Una vez finalizada la codificación se ejecuta el archivo mediante el comando **gcc caso_2.c -o caso_2**; posterior a esto se diseña el entorno de interfaz de líneas de comandos que facilitará



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

la visualización de la práctica. El entorno consiste en cuatro interfaces en las cuales cada una ejecutará un comando/instrucción diferente.

En la segunda interfaz se ejecuta el proceso, con: **./caso_2**

Una vez realizado esto, en la tercera interfaz se ejecuta: **ps -a**, en la que se visualiza el PID del proceso caso_2, en este caso: 4126.

En la cuarta interfaz se ejecuta el comando: **pgrep caso_2**, donde aparece específicamente el PID del caso_2 (caso contrario a **ps -a** donde se listaban todos los procesos en ejecución)

Finalmente en la primera interfaz se ejecuta el comando **htop** para monitorear visualmente el proceso, encontrándose este al final de la interfaz junto con sus características y consumos.

Conclusión

Se ha finalizado la práctica, obteniendo los resultados esperados correctamente. Se ha verificado que el proceso que asigna 400 MB de memoria y ejecuta un bucle infinito para simular una carga constante en el sistema dentro de la terminal visual (generada por el comando **htop**) ocupa 402MB de memoria, 2 MB más de la asignada dentro del programa realizado en C.

Esta práctica no solo familiariza al estudiante a retener comandos básicos de Linux como: nano, htop, ls, ps -a, pgrep, etc. Y por otro lado se visualiza el consumo de memoria que tiene un proceso y como se visualiza en la terminal **htop** en Linux. Entendiendo de esta manera la forma en la que funciona la gestión de memoria en un sistema operativo.