

Para resolver las siguientes preguntas, debe apoyarse en la guía de laboratorio proporcionada. Esta guía permitirá identificar qué archivos utilizar, qué comandos ejecutar y los pasos que se deben seguir en cada actividad para llegar a la solución de las preguntas planteadas.

ACTIVIDAD 1. EXPLORANDO LAS SECCIONES DE UN PROGRAMA EN C

1. Use el comando ***objdump*** para explorar la sección ***.text*** del ejecutable. Luego, identifique una línea en su programa en C y su traducción a ASM en el ejecutable.

```
// .stack → Variable local
int local_var = 7;
```

En la sección ***.text*** del ejecutable aparece traducida como se muestra a continuación.

1d:	c7 45 f4 07 00 00 00	mov	DWORD PTR [ebp-0xc],0x7
-----	----------------------	-----	-------------------------

Usando el siguiente prompt en ChatGPT

Decodifica la instrucción en IA32 "c7 45 f4 07 00 00 00" mostrado los campos en una tabla de forma compacta.
--

La IA va a mostrar la instrucción decodificada, indicando el formato de instrucción (fila 1), el operando que representa en hexa el modo de direccionamiento del primer operando (fila 2), el desplazamiento (fila 3) y el segundo operando que es inmediato (fila 4).

Byte(s)	Campo	Valor / Significado
C7	Opcode	MOV r/m32, imm32
45	ModR/M	mod=01, reg=000, r/m=101 (EBP)
F4	Displ	-12
07 00 00 00	Immediate	7

Complete los pasos para la siguiente línea de código en C del programa.

*heap_var = 99;

En la sección ***.text*** del ejecutable aparece traducida como se muestra a continuación.

TODO...

Usando el siguiente prompt en ChatGPT

TODO...

La IA va a mostrar la instrucción decodificada.

TODO...

2. Use el comando ***objdump*** para explorar la sección ***.data*** del ejecutable. Luego, elabore un prompt en ChatGPT que explique el contenido de la sección. Finalmente, determine a qué sección del código en C corresponde el valor “2a000000” reflejado en ***.data***. Documente el prompt e indique la línea de código relacionada.

Prompt

```
# TODO...
```

Línea de código relacionada.

```
# TODO...
```

Explique con sus propias palabras y de forma breve, qué representa “2a000000” en el programa.

TODO...

3. Use el comando ***objdump*** para explorar la sección ***.bss*** del ejecutable. Luego, elabore un prompt en ChatGPT que explique el contenido de la sección. Finalmente, determine a qué sección del código en C corresponde la información reflejada en ***.bss***. Documente el prompt e indique la línea de código relacionada.

Prompt

```
# TODO...
```

Línea de código relacionada.

```
# TODO...
```

Explique con sus propias palabras y de forma breve, qué información se guarda en el la sección ***.rodata***

TODO...

4. Use el comando ***objdump*** para explorar la sección ***.rodata*** del ejecutable. Luego, elabore un prompt en ChatGPT que explique el contenido de la sección. Finalmente, determine a qué sección del código en C corresponde la información reflejada en ***.rodata***. Documente el prompt e indique las líneas de código relacionadas.

Prompt

```
# TODO...
```

Línea de código relacionada.

```
# TODO...
```

Explique con sus propias palabras de forma breve, qué información se guarda en la sección **.rodata**

TODO...

ACTIVIDAD 2. DE PROGRAMA A PROCESO: EL CAMINIO HACIA LA EJECUCIÓN

1. Use el comando **strace** para interceptar las llamadas al sistema que se llevan a cabo durante el proceso de ejecución de un programa. Luego, identifique la llamada al sistema **execve**. Use el siguiente prompt para entender el formato de la llamada al sistema.

Prompt

Explicame el formato de esta llamada al sistema en una tabla de forma compacta

```
execve("./program", ["/./program"], 0x7ffd997dcdd0 /* 44 vars */) = 0 [ Process PID=1060524 runs  
in 32 bit mode. ]
```

Tabla, resultado de la consulta.

TODO...

Si el programa se ejecuta de la siguiente manera: **strace ./program param_1 param_2**, explique brevemente con sus propias palabras: ¿cuál sería el valor del segundo parámetro en la llamada al sistema **execve**? ¿Por qué es útil que un programa pueda recibir argumentos desde la línea de comandos?

TODO...

2. Realice el procedimiento anterior, pero para la llamada al sistema **write**. Complete los campos y responda la pregunta relacionada

Prompt

TODO...

Tabla, resultado de la consulta.

TODO...

Explique brevemente con sus propias palabras: ¿qué hace la llamada al sistema **write**? ¿Qué función en el programa de usuario (código C) hace el llamado a **write**?

TODO...

3. Use `cat /proc/<PID>/status` para inspeccionar información del ***Process Control Block (PCB)*** de un proceso. Cree una tabla que muestre los campos del PCB que indican: memoria virtual asignada, memoria swap usada, número de hilos activos, CPUs disponibles para el proceso, cambios de contexto ocurridos durante la ejecución y el estado del proceso.

Completar la siguiente tabla

Campo	Valor del proceso	Descripción breve
Memoria virtual asignada		Cantidad de memoria virtual usada por el proceso
Memoria swap usada		Cantidad de memoria swap utilizada por el proceso
Número de hilos activos		Número de threads que ejecuta el proceso
CPUs disponibles		CPUs en las que el proceso puede ejecutarse
Cambios de contexto		Número de context switches ocurridos durante la ejecución
Estado del proceso		R = Ejecutando, S = Durmiendo, D = Espera ininterrumpible, T = Detenido, Z = Zombie

4. Use `cat /proc/<PID>/maps` para inspeccionar el mapa de memoria un proceso. Complete la siguiente tabla indicando, los rangos de direcciones en memoria virtual asignados para cada uno de los segmentos del programa.

Completar la siguiente tabla

Rango	Permisos	Región	Contenido típico
	r--p	.text	Código ejecutable del programa
	rw-p	.data / .bss	Variables globales y estáticas
	rw-p	[heap]	Memoria dinámica (malloc, calloc)
	rw-p	[stack]	Variables locales, parámetros y dirección de retorno

Responda las siguientes preguntas

- ¿Cuántos bytes tiene asignados la sección stack en memoria virtual del proceso?
- ¿Cuántos bytes tiene asignados la sección heap en memoria virtual del proceso?
- ¿Qué implicaciones o limitaciones imponen estos tamaños sobre la ejecución del programa, en particular respecto a:
 - Llamadas a procedimientos y profundidad de la pila (stack).
 - Creación de variables dinámicas en tiempo de ejecución (heap).