


	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Sistemas operativos				
TÍTULO DE LA PRÁCTICA:	<i>Programacion en C y C++ Usando el terminal linux</i>				
NÚMERO DE PRÁCTICA:	<i>02</i>	AÑO LECTIVO:	<i>2023</i>	NRO. SEMESTRE:	<i>05</i>
FECHA DE PRESENTACIÓN	<i>23/05/2023</i>	HORA DE PRESENTACIÓN			
INTEGRANTE (s): Yoset Cozco Mauri				NOTA:	
DOCENTE(s): <i>NIETO VALENCIA, RENE ALONSO</i>					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>1. Se deberá de probar, compilar y ejecutar los siguientes códigos:</p> <p style="margin-left: 20px;">a. Código 01:</p>

```
1 //CREAR ARCHIVO proceso1.c
2 #include <sys/types.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6 int main(void)
7 {
8     pid_t pid;
9     /* fork a child process */
10    pid = fork();
11    if (pid < 0) { /* error occurred */
12        fprintf(stderr, "Fork Failed");
13        return 1;
14    }
15    else if (pid == 0) { /* child process */
16        execlp("/bin/ls", "ls", NULL);
17    }
18    else { /* parent process */
19        /* parent will wait for the child to complete */
20        //wait(NULL);
21        printf("Child Complete");
22    }
23    return 0;
24 }
```

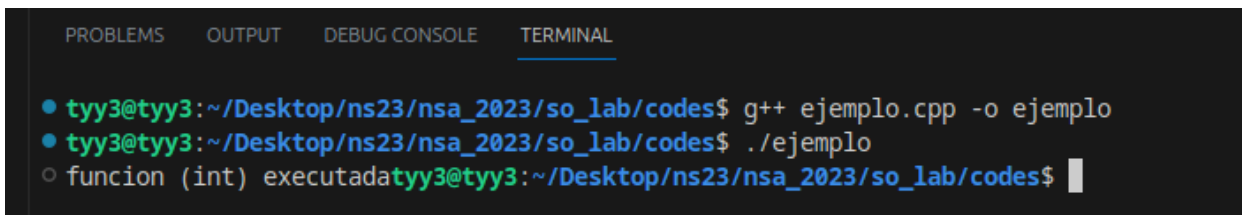
Prueba de código:

```
• tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ gcc proceso1.c -o proceso01
• tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ ./proceso01
Child Completeejemplo.cpp      LinkedList.h  ListNode.h  Main.cpp   proceso1.c
LinkedList.cpp  ListNode.cpp list.txt   proceso01 proceso.c
○ tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ █
```

B. Código 02:

```
1 //CREAR ARCHIVO ejemplo.cpp
2 #include <iostream>
3 class Laboratorio
4 {
5     int num;
6 };
7 class Practica
8 {
9     int a;
10    Laboratorio lab;
11
12    public:
13    operator Laboratorio () { return lab; }
14    operator int () { return a; }
15 };
16 void funcion ( int a) {
17     std::cout << "funcion (int) ejecutada";
18 }
19 void funcion ( Laboratorio la ) {
20     std::cout << "funcion (Laboratorio) ejecutada";
21 }
22 int main()
23 {
24     Practica p;
25     funcion(p);
26     return 0;
27 }
```

Prueba de código: se realizo el cambio de la funcion linea 16 en el parámetro de `int a > Practica a`



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
• tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ g++ ejemplo.cpp -o ejemplo
• tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ ./ejemplo
○ funcion (int) ejecutada tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$
```

C. Código 03:

```
1 //CREAR ARCHIVO LinkedList.h
2 #pragma once
3
4 #include "ListNode.h"
5 #include <iostream>
6 class LinkedList
7 {
8     private:
9         ListNode* _phead;
10
11     public:
12         LinkedList();
13         void insert(int n);
14         void print(void);
15
16         void deleteAll(void);
17         void deleteNodes(ListNode* pn);
18 };
```

Prueba de código: La ejecución independiente del archivo genera el error

```
tyy3@tyy3:~/Desktop/ns23/nsa_2023/so_lab/codes$ g++ LinkedList.cpp -o LinkedList
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/12/../../../../x86_64-linux-gnu/Scrt1.o: in function `_start':
(.text+0x1b): undefined reference to `main'
/usr/bin/ld: /tmp/ccqpxJwR.o: in function `LinkedList::insert(int)':
LinkedList.cpp:(.text+0x55): undefined reference to `ListNode::ListNode()'
/usr/bin/ld: LinkedList.cpp:(.text+0x88): undefined reference to `ListNode::ListNode()'
/usr/bin/ld: /tmp/ccqpxJwR.o: in function `LinkedList::deleteNodes(ListNode*)':
LinkedList.cpp:(.text+0x26e): undefined reference to `ListNode::~~ListNode()'
collect2: error: ld returned 1 exit status
```

D. Código 04:

```
1 //CREAR ARCHIVO LinkedList.cpp
2 #include "LinkedList.h"
3 LinkedList::LinkedList() {
4     _phead = nullptr;
5 }
6 void LinkedList::insert(int n) {
7     if (_phead == nullptr) {
8         _phead = new ListNode();
9         _phead->_value = n;
10        return;
11    }
12    else {
13        ListNode* pn = new ListNode();
14        pn->_value = n;
15        ListNode* pnode = _phead;
16        while (pnode->_pNext != nullptr && pnode->_pNext->_value < n) {
17            pnode = pnode->_pNext;
18        }
19        if(pnode->_pNext != nullptr && pnode->_value > n){
20            pn->_pNext = _phead;
21            _phead = pn;
22        }
23        if (pnode->_pNext == nullptr) {
24            pnode->_pNext = pn;
25        } else {
26            pn->_pNext = pnode->_pNext;
27            pnode->_pNext = pn;
28        }
29    }
30 }
31 void LinkedList::print(void) {
32     ListNode* pnodes = _phead;
33     while (pnodes->_pNext != nullptr) {
34         std::cout << pnodes->_value << " ";
35         pnodes = pnodes->_pNext;
36     }
37 }
38 void LinkedList::deleteAll(void) {
39     if (_phead != nullptr) {
40         deleteNodes(_phead);
41     }
42 }
43 void LinkedList::deleteNodes(ListNode *pn) {
44     if (pn->_pNext != nullptr) {
45         deleteNodes(pn->_pNext);
46     }
47     delete(pn);
48 }
```

```
1 //CREAR ARCHIVO ListNode.cpp
2 #include "ListNode.h"
3 ListNode::ListNode() {
4     _value = -1;
5     _pNext = nullptr;
6 }
7 ListNode::~~ListNode() {
8     //delete _pNext;
9 }

1 //CREAR ARCHIVO ListNode.h
2 #pragma once
3 class ListNode {
4     public:
5     int _value;
6     ListNode* _pNext;
7
8     ListNode();
9     ~ListNode();
10 };

1 //CREAR ARCHIVO main.cpp
2 #include <iostream>
3 #include <stdlib.h>
4 #include "LinkedList.h"
5
6 int main() {
7     int max = 10;
8     LinkedList* plist = new LinkedList();
9
10    for (int i = 0; i < max; i++) {
11        int num = rand()%max;
12        plist->insert(num);
13    }
14    plist->print();
15    plist->deleteAll();
16    delete(plist);
17
18    return 0;
19 }
```

Prueba de código: se ejecuta el siguiente comando que combina los archivos necesarios: g++
LinkedList.cpp ListNode.cpp Main.cpp -o programa

```
• tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ g++ LinkedList.cpp ListNode.cpp Main.cpp -o programa
• tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$ ./programa
o 1 3 5 6 7 tty3@tty3:~/Desktop/ns23/nsa_2023/so_lab/codes$
```

II. SOLUCIÓN DEL CUESTIONARIO

¿Cuál es la diferencia entre compilar con GCC y G++? Cada uno esta orientado a compilar diferentes archivos GCC compila de C, claro que también puede compilar archivos de c++ pero las librerías no estan incluidas por defecto, el cual debe ser agregado manualmente . En cambio si usamos g++ para compilar programas en c++ las bibliotecas de c++ vienen por defecto.


¿En qué se diferencia el archivo generado “.o” contra un “.exe”?

El archivo “.o” es un archivo intermedio para crear un archivo ejecutable “.exe” , el archivo objeto contiene código compilado y símbolos, pero no es un archivo ejecutable por si mismo, en cambio el archivo “.exe” es el resultado final de la compilación y enlazado de varios archivos “.o”/

III. CONCLUSIONES

En conclusión, los archivos objeto (.o) son generados al compilar archivos fuente utilizando un compilador como gcc o g++. Estos archivos objeto contienen el código de máquina compilado y símbolos utilizados para enlazar el programa final. Sin embargo, los archivos objeto por sí solos no son ejecutables, ya que les falta información necesaria.

RETROALIMENTACIÓN GENERAL

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

<p>REFERENCIAS Y BIBLIOGRAFÍA</p>