Universidad de Antioquia
Facultad de Ingeniería
Sistemas Operativos y Laboratorio

Práctica de Laboratorio # 2 – Manejo dinámico de memoria

Optimización de código en C con GDB y Valgrind

1. Escriba un programa simple llamado null.c que cree un puntero a un entero, llévelo a null y entonces intente des-referenciarlo (esto es, asignarle un valor). Compile este programa llamado null. ¿Qué pasa cuando usted ejecuta este programa?
2. Compile el programa del ejercicio anterior usando información de símbolos (flag -g). Al hacer esto se está poniendo más información en el ejecutable para permitir al debugger acceder a información útil sobre los nombres de las variables y cosas similares. Ejecute el programa bajo el debugger digitando en consola (para el caso) gdb null y entonces una vez el gdb este corriendo ejecute run. ¿Qué muestra gdb?
3. Haga uso de la herramienta valgrind en el programa empleado en los puntos anteriores. Se usará la herramienta memcheck que es parte de valgrind para analizar lo que pasa: valgrind --leak-check=yes null. ¿Qué pasa cuando corre esto?, ¿Puede usted interpretar la salida de la herramienta anterior?
4. Escriba un programa sencillo que asigne memoria usando malloc() pero olvide liberarla antes de que el programa termina. ¿Qué pasa cuando este programa se ejecuta?, ¿Puede usted usar gdb para encontrar problemas como este?, ¿Que dice acerca de Valgrind (de nuevo use este con la bandera –leak-check=yes)?
5. Escriba un programa que cree un array de enteros llamado data de un tamaño de 100 usando malloc; entonces, lleve el data[100] a 0. ¿Qué pasa cuando este programa se ejecuta?, ¿Qué pasa cuando se corre el programa usando valgrind?, ¿El programa es correcto?
6. Codifique un programa que asigne un array de enteros (como arriba), luego lo libere, y entonces intente imprimir el valor de un elemento del array. ¿El programa corre?, ¿Qué pasa cuando hace uso de valgrind?
7. Ahora pase un funny value para liberar (e.g. un puntero en la mitad del array que usted ha asignado) ¿Qué pasa?, ¿Usted necesita herramientas para encontrar este tipo de problemas?
8. Intente usar alguna de las otras interfaces para asignación de memoria. Por ejemplo, cree una estructura de datos simple similar a un vector y que use rutinas que usen realloc para manejar el vector. Use un array para almacenar los elementos del vector; cuando un usuario agregue una entrada al vector, use realloc para asignar un espacio

más a este. ¿Qué tan bien funciona el vector así?, ¿Como se compara con una lista enlazada?, utilice valgrind para ayudarse en la búsqueda de errores.

9. Considere el siguiente fragmento de código:

```
char* getString(){
	char message[100] =  "Hello World!";
	return message;
}

int main(int argc, char* argv[]){
	printf("String: %s\n", getString());
}
```

¿Necesita GDB o Valgrind para identificar el error? ¡Explique el porqué de dicho error!

10. En el ejemplo siguiente:

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[]){
	const int NUM_HEIGHTS = 3;
	int *heights = malloc(NUM_HEIGHTS);
	for(int i=0; i<NUM_HEIGHTS; i++){
		heights[i] = i*i;
		printf("%d: %d\n", i, heights[i]);
	}
}
```

¿Encuentra algún error con GDB? ¿alguno con Valgrind? Explique.

11. Analice el siguiente fragmente de código:

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[]){
	const int NUM_HEIGHTS = 3;
	int *heights = malloc(NUM_HEIGHTS);
	for(int i=0; i<NUM_HEIGHTS; i++){
		heights[i] = i*i;
```

```
            printf("%d: %d\n", i, heights[i]);
            free(heights);
        }
    }
```

¿Cuál es el error en el uso de la memoria? ¿Qué salida presenta Valgrind?

12. Identifique y resuelva el problema del siguiente código utilizando las herramientas GDB y Valgrind:

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[]){
    const int NUM_HEIGHTS = 10;
    int *heights = malloc(NUM_HEIGHTS*sizeof(*heights));
    for(int i=0; i<NUM_HEIGHTS; i++){
        if(heights = NULL){
            heights = malloc(NUM_HEIGHTS*sizeof(*heights));
        }
    }
    free(heights);
}
```

13. Utilice las herramientas gdb y Valgrind para optimizar el código fuente de su práctica #1. Corrija todos los problemas de manejo de memoria dinámica señalados por valgrind. Para cada error encontrado indique: ¿En qué consiste el problema? ¿Cómo lo resolvió?
14. Seleccione uno de los siguientes módulos de Valgrind*: cachegrind, callgrind, helgrind, DRD, Massif o Dhat*. Explique las opciones básicas del mismo (las 4 o 5 opciones de uso más importantes y comunes) y presente un ejemplo de su utilización. ¿Para qué escenarios es útil la herramienta que seleccionaste?

Referencias

Manual de Valgrind: https://valgrind.org/docs/manual/manual-core.html#manual-core.report

Manual de GDB: https://linux.die.net/man/1/gdb