Programación de software de sistemas

Ayudantía 6

Profesor: Rodrigo Verschae

Ayudante: Nicolás Araya

Valgrind

Valgrind es un framework utilizado para el análisis de memoria dinámica. Tiene herramientas que pueden detectar automáticamente bugs en hilos y memory leaks.

Instalación (Linux):

```
sudo apt-get install valgrind
```

https://valgrind.org/

Valgrind

Modo de uso:

```
gcc -g -o myprog myprogram.c // Output en modo debuggin

valgrind --leak-check=yes ./myprog arg1 arg2
```

https://valgrind.org/docs/manual/quick-start.html

Code 1

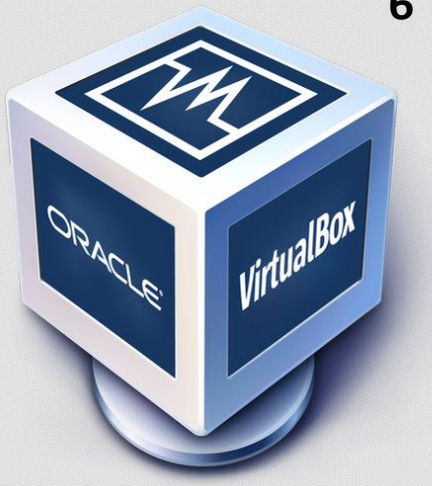
```c
#include <stdlib.h>

 void f(void)
 {
    int* x = malloc(10 * sizeof(int));
    x[10] = 0;        // problem 1: heap block overrun
 }                    // problem 2: memory leak -- x not freed

 int main(void)
 {
    f();
    return 0;
 }
```

```
==19182== Invalid write of size 4
==19182==    at 0x804838F: f (example.c:6)
==19182==    by 0x80483AB: main (example.c:11)
==19182==  Address 0x1BA45050 is 0 bytes after a block of size 40 alloc'd
==19182==    at 0x1B8FF5CD: malloc (vg_replace_malloc.c:130)
==19182==    by 0x8048385: f (example.c:5)
==19182==    by 0x80483AB: main (example.c:11)
```

PID

Ubuntu

Problema, termina el código.



En una verdulería se está implementando el sistema de la cinta para la venta de frutas a mitad de precio. Este sistema consiste en que van colocando frutas 1 por 1 en una cinta y los clientes las recoge en el mismo orden en que fueron puestas.

- Hay 4 distintos tipos de fruta
- Caben n frutas en la cinta.
- **No pueden colocar más fruta mientras la cinta esté llena.**
- Siempre deben colocar la fruta en la posición siguiente (debe estar vacía)

Estructuras y variables globales

```
int FRUIT[4];
int TOTAL;
```

```
typedef struct{
    int id;
    int price;
    char *name;
} Product;
```

```
typedef struct{
    int in, out, size, cnt, end;
    Product **array;
} Buffer;
```

Auxiliares

```
void put(Buffer *buf, Product *product){

    while(buf->size == buf->cnt)

    buf->array[buf->in]=product;
    buf->in=(buf->in+1)%(buf->size);
    buf->cnt++;


}
```

```
Product *get(Buffer *buf){
   Product *product;


   while(buf->cnt == 0){
      if (buf->end == 1){



         return NULL;}


   }
   product = buf->array[buf->out];
   buf->out = (buf->out+1)%(buf->size);
   buf->cnt--;



   return product; }
```

```
Buffer *create_buffer(int size){
    Buffer *buf = (Buffer*) malloc(sizeof(Buffer));
    buf->array = (Product**) malloc(size * sizeof(Product*));
    buf->size = size;
    buf->in = buf->out = buf->cnt = buf->end = 0;


    return buf;
}
```

```
Product *create_product(int id){
    Product *product = malloc(sizeof(Product));
    id = id%4;
    product->id = id;
    product->price = (id + 1)*3;
    if(id == 0) product->name = "Apple";
    else if (id == 1) product->name = "Pear";
    else if (id == 2) product->name = "Banana";
    else if (id == 3) product->name = "Kiwi";
    return product;
}
```

```
void *producer(void* ptr){
    Buffer *buf = (Buffer*) ptr;
    for(int i = 0 ; i < 100000; i++){
        Product *product = create_product(i);
        put(buf, product);
    }

    buf->end = 1;



    return NULL;
}
```

```
void update_information(Product *producto, int *fruit){

    FRUIT[product->id]++;
    TOTAL++;

}
```

```
void* consumer(void* ptr){
    Buffer *buf = (Buffer*) ptr;

    int total_price = 0;
    for(;;){
        Product *product = get(buf);
        if (product == NULL)
            break;
        update_information(product);
        total_price=total_price+(product->price);
    }
    printf("total price: \t$%d\n", total_price);
    return NULL;}
```

```
int main(){

    TOTAL = 0;
    for(int i = 0; i < 4; i++)
        FRUIT[i]=0;
    pthread_t t_factory, t_client1, t_client2;
    int BUFF_SIZE = 10;
    Buffer *buf = create_buffer(BUFF_SIZE);

    pthread_mutex_init(&mtx_cnt, NULL);




}
```

Solución

https://github.com/NicolasAraya932/AyudantiaPSS

Programación de software de sistemas

Ayudantía 6

Profesor: Rodrigo Verschae

Ayudante: Nicolás Araya