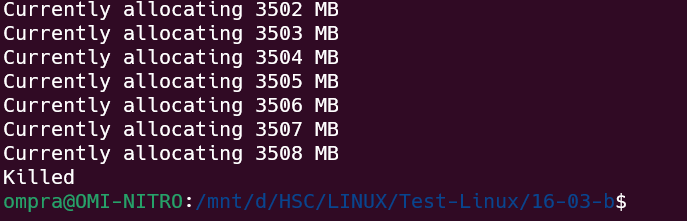
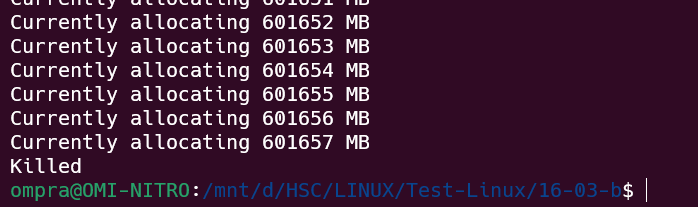
Heapfiller.c

When we just allocate the memory and do not use it so malloc function runs for allocation of more megabytes of memory.

When we do memset we actually use it.

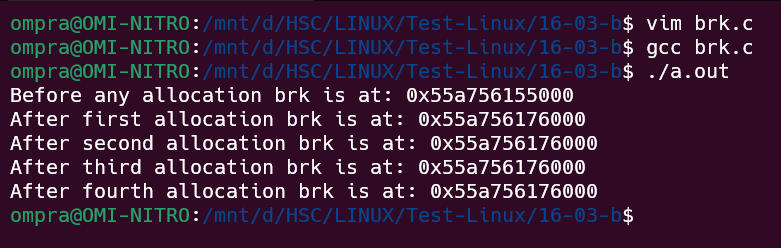
It is the optimistic memory allocation scheme of Linux OS.



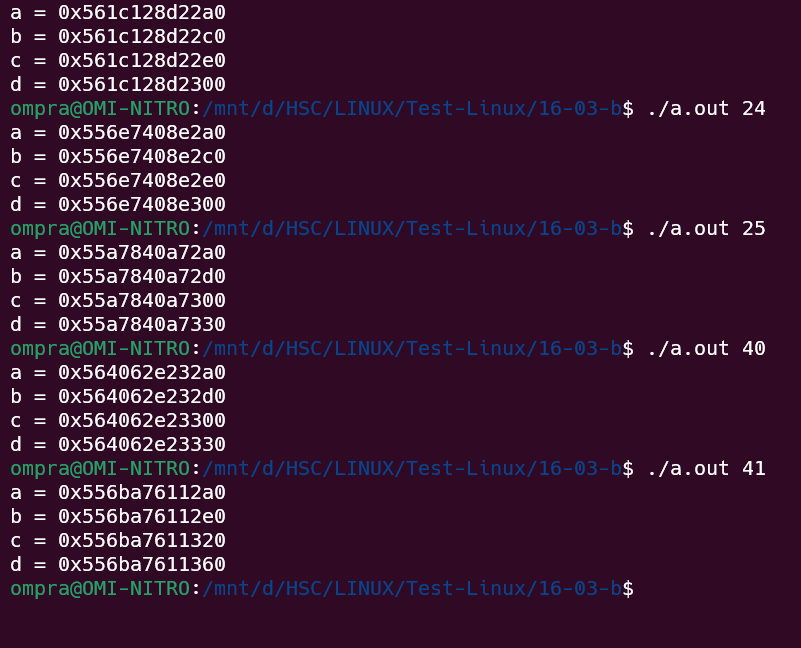


brk.c

when malloc allocates some memory on heap, brk moves to little more than the allocated size(actual).So, after the first alloc, the brk remains constant for furthut mallocs.



Allocated\_block.c



When we allocate 1-24 byte 32 bytes space is allocated.

When we allocate 24-40 bytes 48 bytes space is allocated.

It happens because

Structure of allocated block is len+mem so len is 8 bytes and for 1 byte mem is 1 byte so total 8 + 1 = 9 bytes, but the minimum memory that is allocated on heap is 32 bytes.

So for 1-24 🡺 1 + 8 --- 24 + 8 id less than equal to 32 so 32. Similarly for others.

Splint -> statically checking C programs.

Hello\_bug.c

ERR-1 => Memory is used after being released.

/\*

\* Video Lecture: 10

\* Programmer: Arif Butt

\* Course: System Programming with Linux

\*/

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

extern int errno;

int main(){

char \* arr = (char\*) malloc (sizeof(char)\*10);

if(arr==NULL){

perror("malloc failed:");

exit(errno);

}

arr[0]='a';

arr[1]='r';

arr[2]='i';

arr[3]='f';

arr[4]='\0';

printf("%s\n",arr);

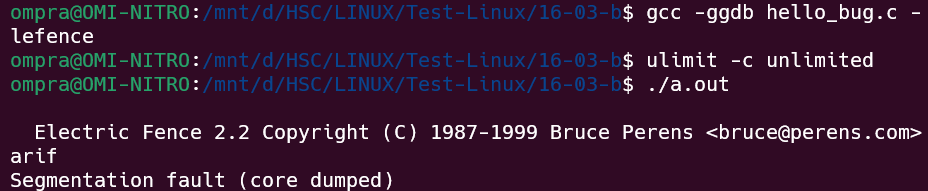
free(arr);

arr[0] = 'b'; //Error

return 0;

}

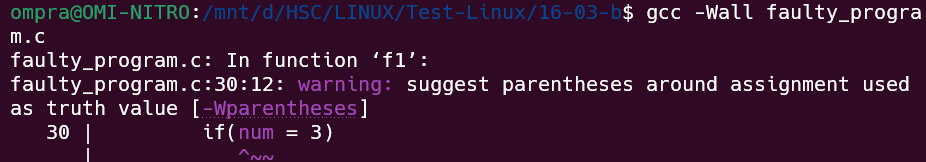
Electric fence



Valgrind – faulty\_program.c

Errors:

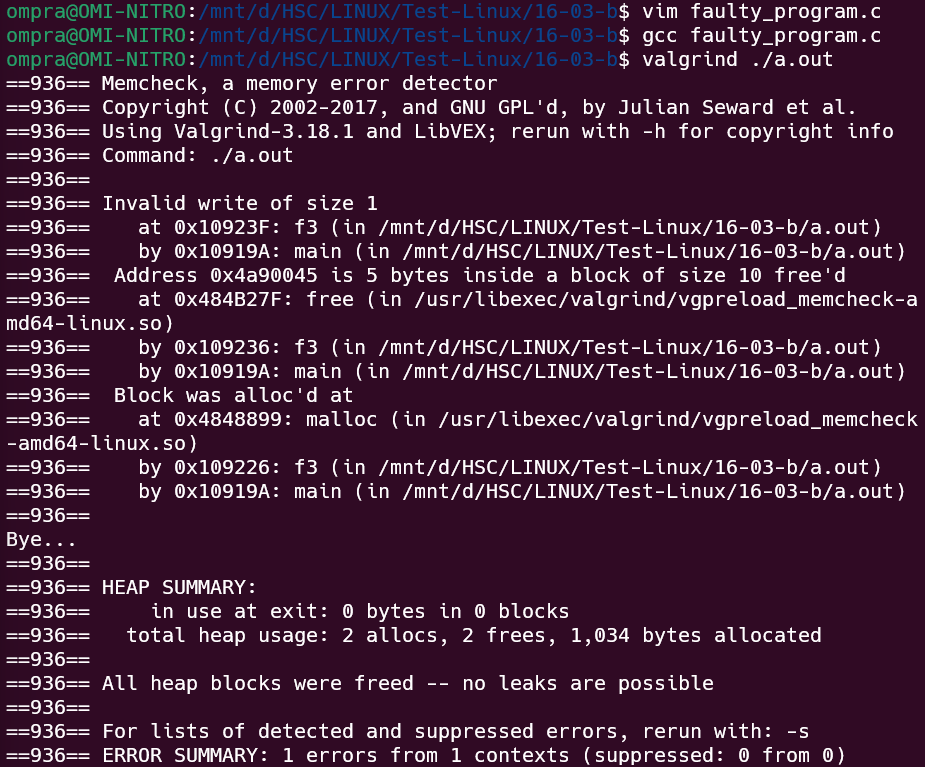
-use of assignment operator in if statement (Programming error which don’t show in normal compilation)



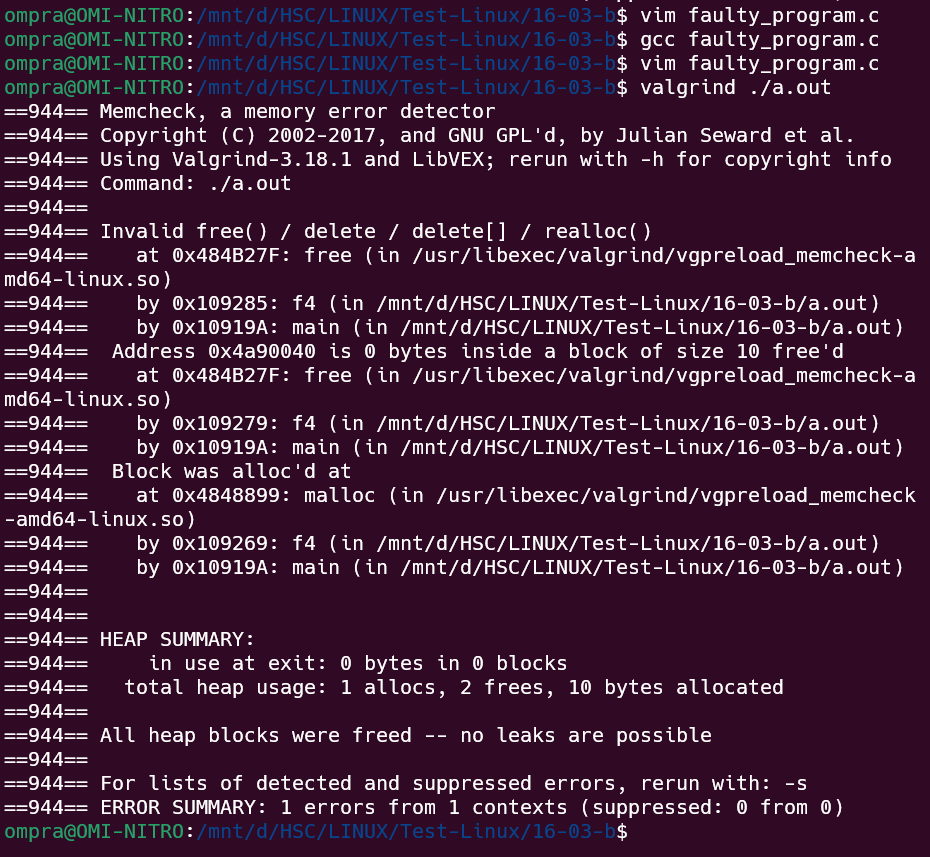
-Using space more than the allocated space.



-Using space after freeing the memory



-Freeing a already freed memory pointer.



-Not freeing the memory

