**Memory leaks and errors practice session**

Typographical conventions

We use the following conventions in this guide:

emacs The name of a specific command or file

file You should replace file with a specific name

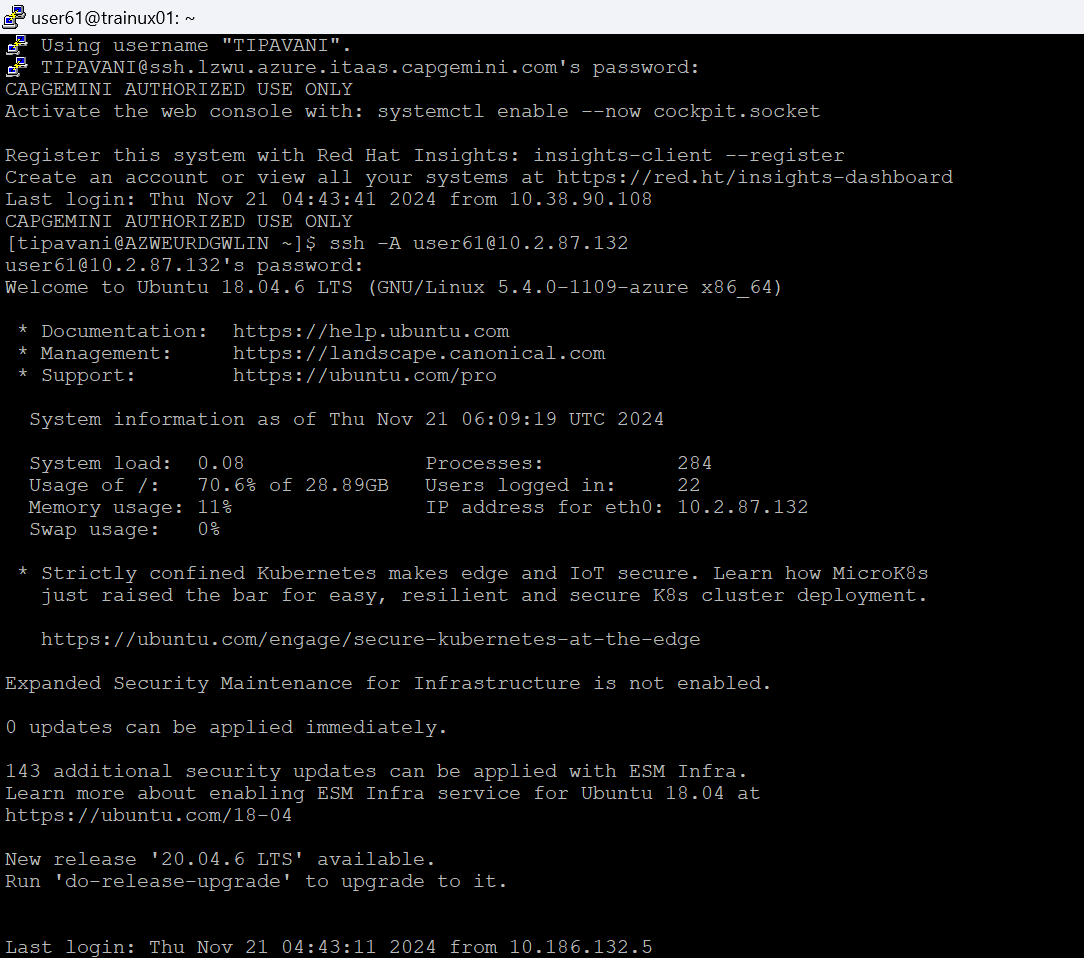
Exit abc Output that you see on the screen

**valgrind command**

valgrind --tool=memcheck –-leak-check=yes ./sample1

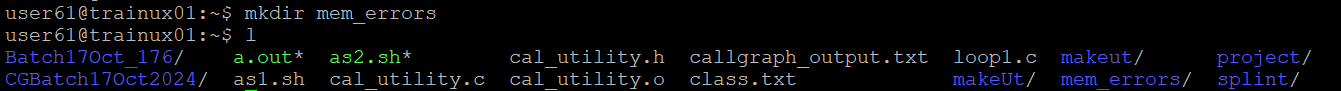
valgrind –-leak-check=yes ./sample1

1. Login into the Linux server with your login Ids



1. Create a new directory called mem\_errors in your home directory <home>

mkdir mem\_errors

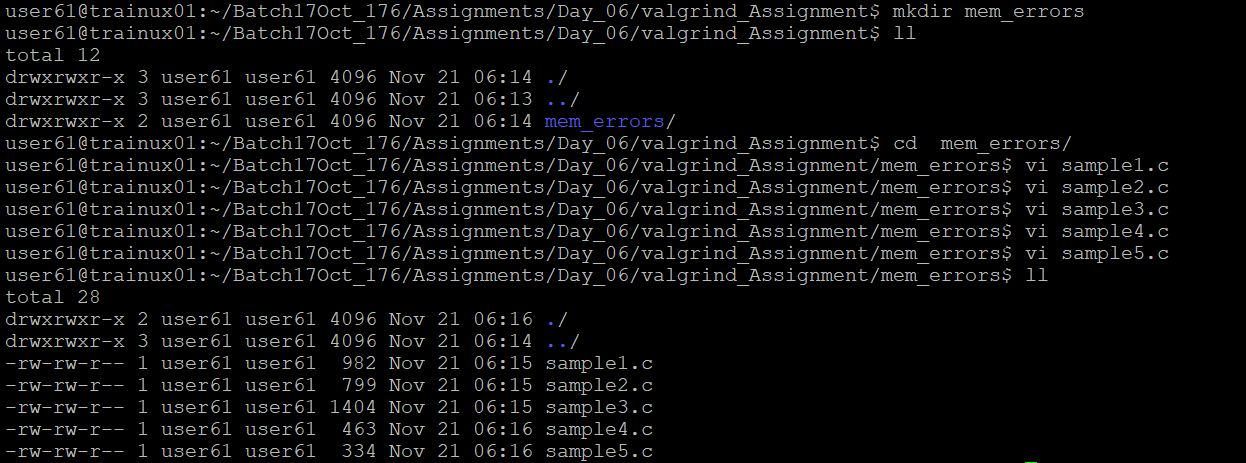


1. Go inside the directory you have created in (2) /<home>/mem\_errors

cd mem\_errors



1. Copy the following files from the path as mentioned by the trainer:
   1. sample1.c
   2. sample2.c
   3. sample3.c
   4. sample4.c
   5. sample5.c



1. Take a look at the example programs and observe if you can find any errors.

Compilation

1. Compile the files

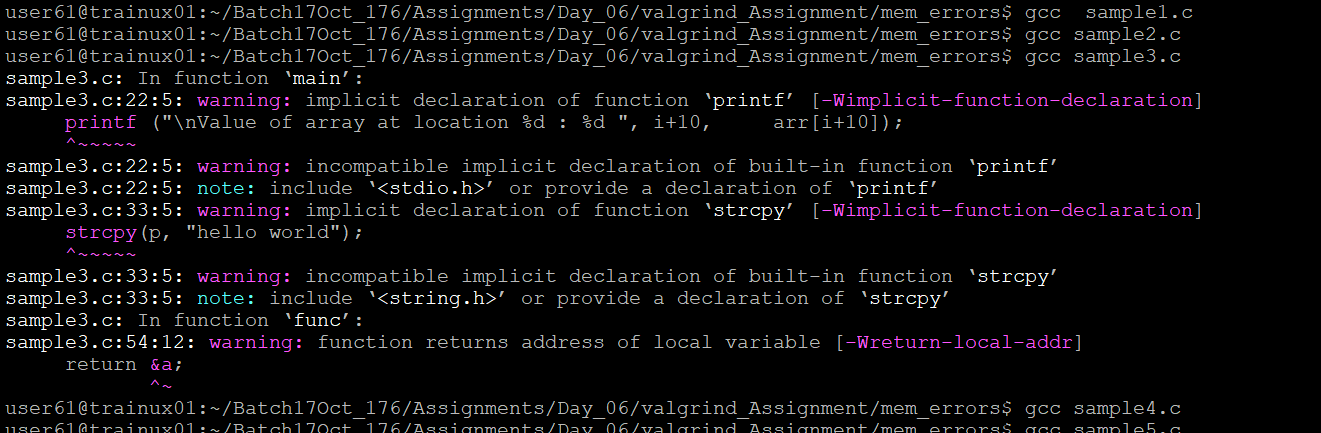
gcc –o sample1 –g sample1.c

gcc –o sample2 –g sample2.c

gcc –o sample3 –g sample3.c

gcc –o sample4 –g sample4.c

gcc –o sample5 –g sample5.c



Execution

1. Execute the file sample1

valgrind –-leak-check=yes sample1

After quitting the program Valgrind displays a list of memory errors in the program. Analyze these errors.

How to analyze the errors given by Valgrind:

For each error, Valgrind gives the back trace of the process starting from the statement at which the error has occurred e.g.

First line of each error is the error type

Invalid read of size 1-----------------à Error type

A screenshot of a computer program

Description automatically generated

The error type is then followed by the back trace of the error

at 0x49064F2: strlen (mac\_replace\_strmem.c:243)

by 0x3585442B4A: vfprintf (in /lib64/tls/libc-2.3.4.so)

by 0x3585448037: printf (in /lib64/tls/libc-2.3.4.so)

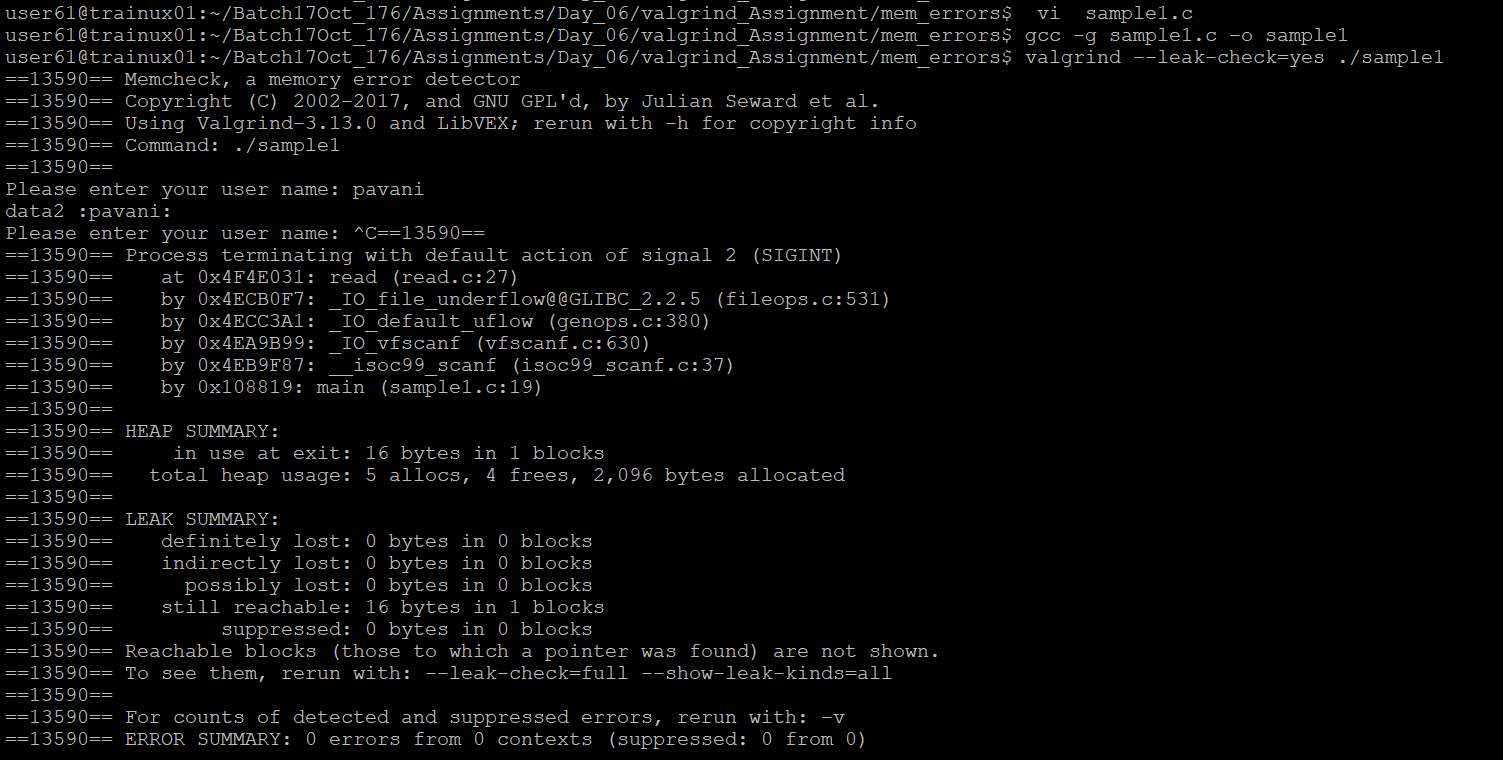
by 0x4006BA: main (sample2.c:37)--------------à Source code statement resulting in the error

Then comes any additional information e.g. if the invalid read error is because we are reading memory which is already freed, then the statement at which free was called for this memory will be shown

Address 0x4A2C030 is 0 bytes inside a block of size 16 free'd

at 0x490555D: free (vg\_replace\_malloc.c:235)

by 0x40068B: main (sample2.c:31)



Please note that Valgrind may display many errors corresponding to a single source code statement. This is because one source code statement may lead to multiple instructions, with each instruction resulting in an invalid memory access. Once the source code is corrected, all the errors will go away.

**Memory Leaks interpretation**

For each memory leak, the backtrace of the statement where the memory was allocated is shown e.g.

32 bytes in 2 blocks are definitely lost in loss record 2 of 2

at 0x4904A06: malloc (vg\_replace\_malloc.c:149)

by 0x40068E: main (sample1.c:30)

2 blocks above signify that the same malloc was called twice and in totality 32 bytes of memory were allocated

Valgrind shows the following categories for each block of memory which is not freed at the end of the process:

* Definitely lost, or "leaked": No pointer to the block can be found. The block is classified as "leaked", because the programmer could not possibly have freed it at program exit, since no pointer to it exists. This is likely a symptom of having lost the pointer at some earlier point in the program.
* Possibly lost: A pointer to the interior of the block is found. The pointer might originally have pointed to the start and have been moved along, or it might be

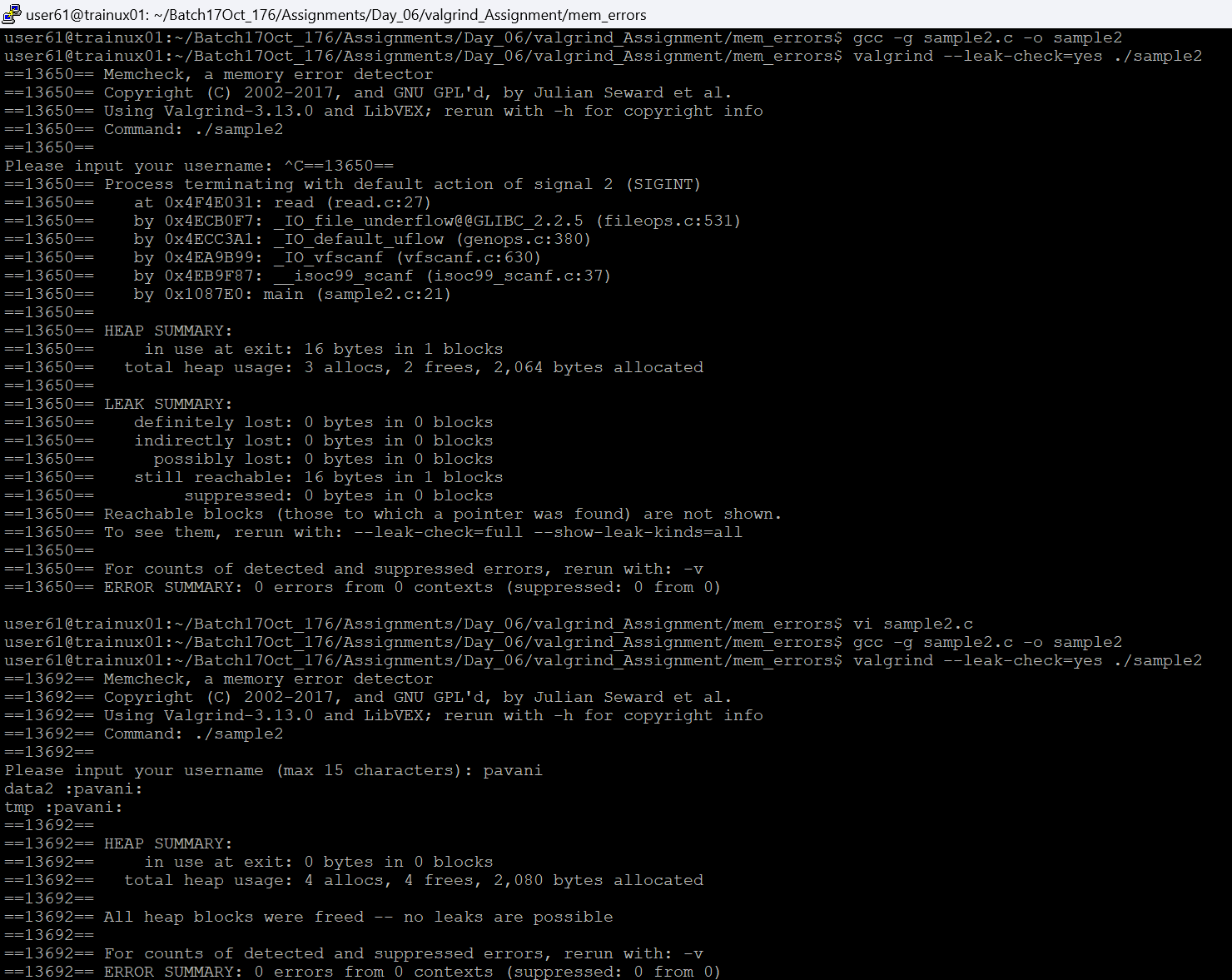
entirely unrelated. Memcheck deems such a block as "dubious", because it's unclear whether or not a pointer to it still exists.

* Still reachable: A pointer to the start of the block is found.

1. Execute the file sample2

valgrind –-leak-check=yes ./sample2

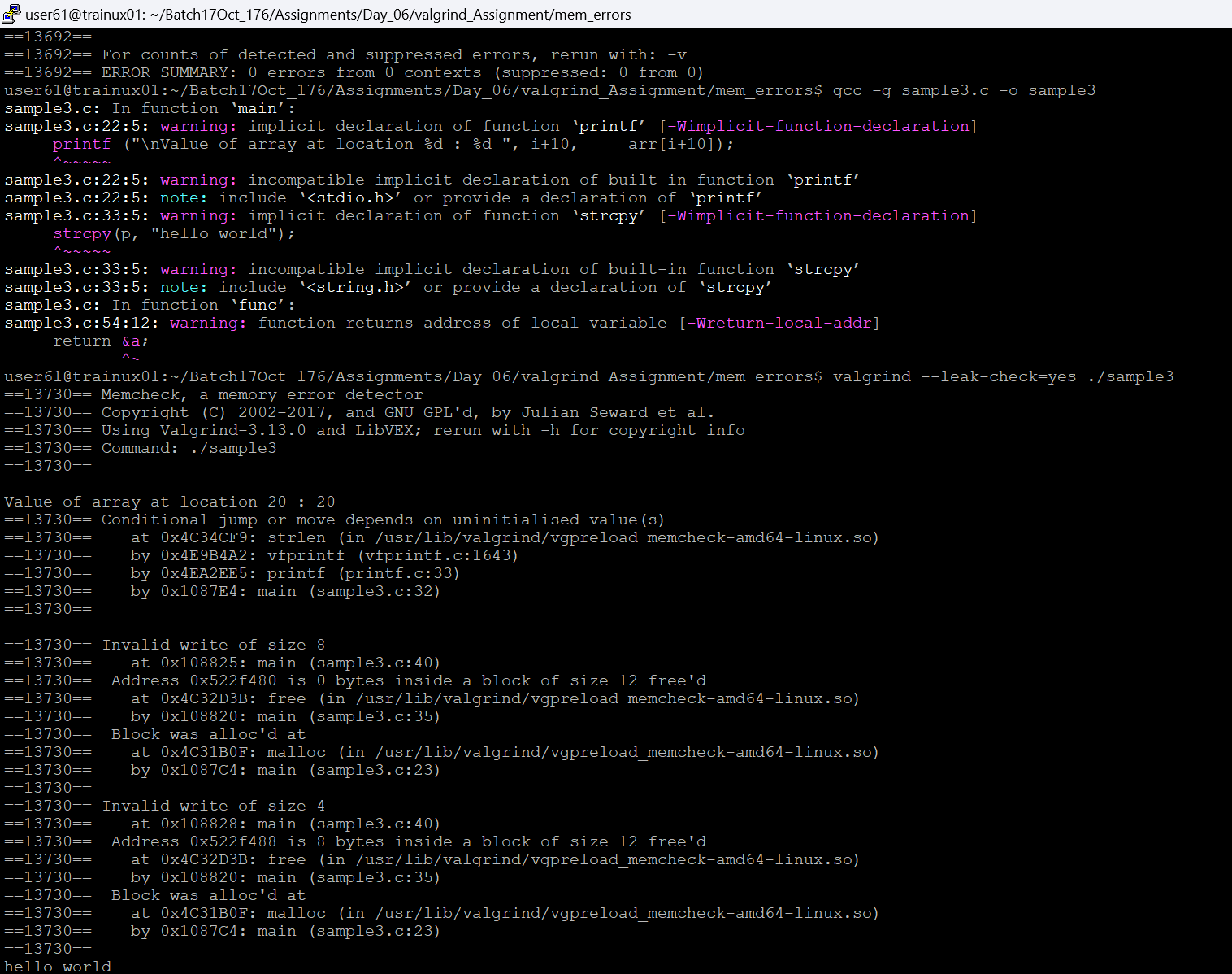
After quitting the program Valgrind displays a list of memory errors in the program. Analyze these errors.



1. Execute the file sample3

Valgrind –-leak-check=yes ./sample3

After quitting the program Valgrind displays a list of memory errors in the program. Analyze these errors.



A computer screen shot of a black screen

Description automatically generated

1. Execute the file sample4

Valgrind –-leak-check=yes ./sample4

After quitting the program Valgrind displays a list of memory errors in the program. Analyze these errors.

A screenshot of a computer program

Description automatically generated

A black screen with white text

Description automatically generated

1. Execute the file sample4

Valgrind –-leak-check=yes ./sample5

After quitting the program Valgrind displays a list of memory errors in the program. Analyze these errors.

A computer screen with many small text

Description automatically generated with medium confidence

A computer screen with text and numbers

Description automatically generated

1. Compile and execute the program in queue\_linked\_list\_memory\_leak

valgrind –leak-check=yes ./queue\_llist

After quitting the program Valgrind displays a list of memory errors in the program. Analyze these errors.

1. Refer the code in “Doublepointer\_Memoryleak”. Fix the memory leaks in the code.