# Valgrind and C-Strings

KOÇ UNIVERSITY

COMP201 Lab3

**Spring 2024**

# What is Valgrind?


Valgrind is:

- An open-source system memory debugger
- Used for memory error and leak detection
- Also Profiling
- Detect common memory errors in C and C++ programs

# Errors that Valgrind can detect and report:

- **Invalid read/write errors**
  - Reads or writes to a memory address which you did not allocate
- **Use of an uninitialized value**
  - Code uses a declared variable before any kind of explicit assignment
- **Invalid free error**
  - Code attempts to delete allocated memory twice
  - Delete memory that was not allocated

# Memory Errors Vs. Memory Leaks

- **Memory *leaks*:**
  - A program dynamically allocates memory and does not free it
  - won't cause a program to misbehave, crash, or give wrong answers

- **Memory errors:**
  - Is a red alert.
  - Reading uninitialized memory
  - Writing past the end of a piece of memory,
  - Accessing freed memory, etc
  - Can have significant consequences.
  - Memory errors should never be treated casually or ignored

# How to compile:

➢ gcc -g -o Out sample.c

-g  Enabling the Valgrind
Out  Output file
Sample.c  The program for compile

❖ Using -O0 is also a good idea! But…

# Example: Sample.c

- with a memory error and a memory leak.

```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;
}
//
```

# Memory error

- valgrind --tool=memcheck ./out

```
ntofighi21@njt:~/darsi/comp201/lab3$ valgrind --tool=memcheck ./out
==33826== Memcheck, a memory error detector
==33826== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==33826== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==33826== Command: ./out
==33826==
==33826== Invalid write of size 4
==33826==    at 0x10916B: f (sample.c:6)
==33826==    by 0x109180: main (sample.c:11)
==33826==  Address 0x4a57068 is 0 bytes after a block of size 40 alloc'd
==33826==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==33826==    by 0x10915E: f (sample.c:5)
==33826==    by 0x109180: main (sample.c:11)
==33826==
```

➢ **valgrind --tool=memcheck ./out**

```
ntofighi21@njt:~/darsi/comp201/lab3$ valgrind --tool=memcheck ./out
==33826== Memcheck, a memory error detector
==33826== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==33826== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==33826== Command: ./out
==33826==
==33826== Invalid write of size 4
==33826==    at 0x10916B: f (sample.c:6)
==33826==    by 0x109180: main (sample.c:11)
==33826==  Address 0x4a57068 is 0 bytes after a block of size 40 alloc'd
==33826==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==33826==    by 0x10915E: f (sample.c:5)
==33826==    by 0x109180: main (sample.c:11)
==33826==
```

process ID

➢ **valgrind --tool=memcheck ./out**

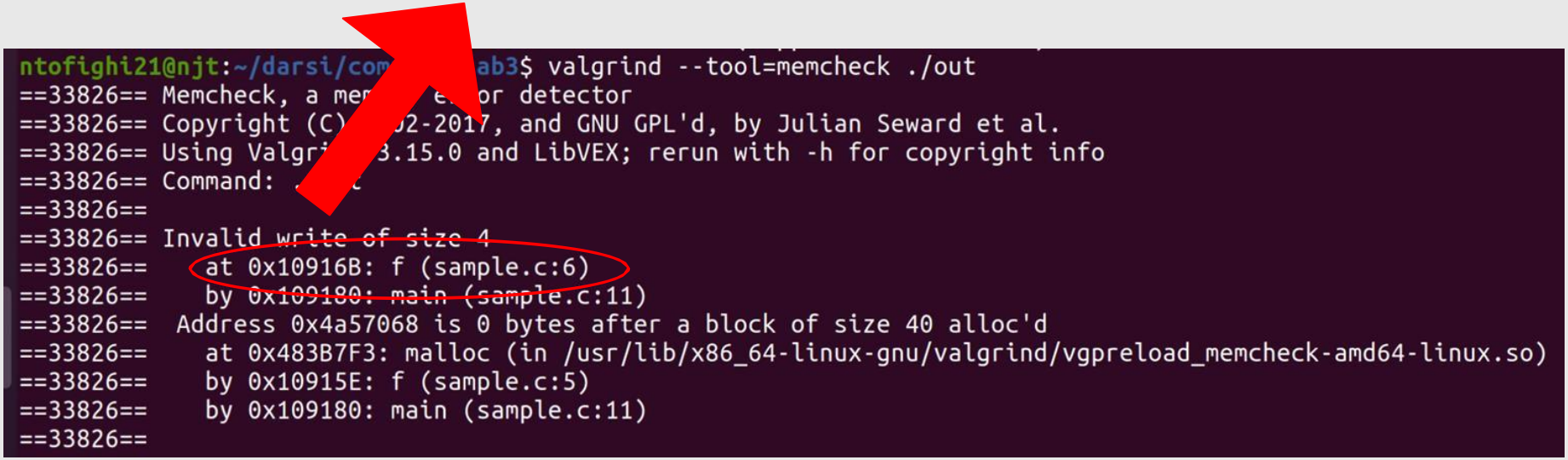
Types of error

Here; The program wrote to some memory it should not have due to a heap block overrun.

```
ntofighi21@n      rsi/comp201/lab3$ valgrind --tool=memcheck ./out
==33826== Mem      a memory error detector
==33826== C      ig    (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==33826==      ng Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==33826==      mmand: ./out
==33826==
==33826== Invalid write of size 4
==33826==    at 0x10916B: f (sample.c:6)
==33826==    by 0x109180: main (sample.c:11)
==33826==  Address 0x4a57068 is 0 bytes after a block of size 40 alloc'd
==33826==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==33826==    by 0x10915E: f (sample.c:5)
==33826==    by 0x109180: main (sample.c:11)
==33826==
```

➢ **valgrind --tool=memcheck ./out**

Stack trace → where the problem occurred.




KOÇ UNIVERSITY

# Memory error

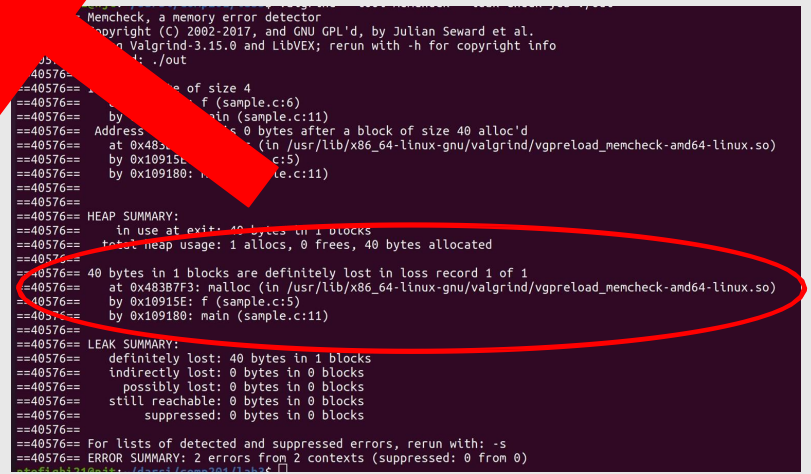
- valgrind --tool=memcheck --leak-check=yes ./out

```
==40576== Memcheck, a memory error detector
==40576== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==40576== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==40576== Command: ./out
==40576==
==40576== Invalid write of size 4
==40576==    at 0x10916B: f (sample.c:6)
==40576==    by 0x109180: main (sample.c:11)
==40576==  Address 0x4a57068 is 0 bytes after a block of size 40 alloc'd
==40576==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==40576==    by 0x10915E: f (sample.c:5)
==40576==    by 0x109180: main (sample.c:11)
==40576==
==40576==
==40576== HEAP SUMMARY:
==40576==     in use at exit: 40 bytes in 1 blocks
==40576==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
==40576==
==40576== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==40576==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==40576==    by 0x10915E: f (sample.c:5)
==40576==    by 0x109180: main (sample.c:11)
==40576==
==40576== LEAK SUMMARY:
==40576==    definitely lost: 40 bytes in 1 blocks
==40576==    indirectly lost: 0 bytes in 0 blocks
==40576==      possibly lost: 0 bytes in 0 blocks
==40576==    still reachable: 0 bytes in 0 blocks
==40576==         suppressed: 0 bytes in 0 blocks
==40576==
==40576== For lists of detected and suppressed errors, rerun with: -s
==40576== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

# Memory error

- valgrind --tool=memcheck --leak-check=yes ./out

```
==40576== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==40576==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprel
oad_memcheck-amd64-linux.so)
==40576==    by 0x10915E: f (sample.c:5)
==40576==    by 0x109180: main (sample.c:11)
==40576==
```



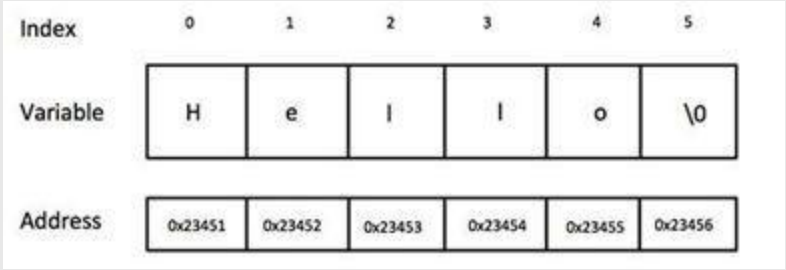
# Strings in C

KOÇ
UNIVERSITY

KOÇ
UNIVERSITY

# C-Strings

- 1-D array of characters
- Terminated by null or \0
- Initializing a String
  - char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
  - char greeting[] = "Hello";



# Standard string functions in C

KOÇ UNIVERSITY

# strcat( )

- Concatenates two given strings.
- Concatenates source string at the end of destination string.
- strcat ( char * destination, char * source );

```c
#include <stdio.h>
#include <string.h>

int main( )
{
   char source[20] = " 201" ;
   char target[20]= " comp" ;
   printf ( "\nSource string = %s", source ) ;
   printf ( "\nTarget string = %s", target ) ;
   strcat(target, source);
   printf ( "\nTarget string after strcat() = %s", target ) ;
}
```

# strcat( )

- Concatenates two given strings.
- Concatenates source string at the end of destination string.
- strcat ( char * destination, char * source );

```c
#include <stdio.h>
#include <string.h>

int main( )
{
   char source[20] = " 201" ;
   char target[20]= " comp" ;
   printf ( "\nSource string = %s", source ) ;
   printf ( "\nTarget string = %s", target ) ;
   strcat(target, source);
   printf ( "\nTarget string after strcat() = %s", target ) ;
}
```

Output:

```
Source string =  201
Target string =  comp
Target string after strcat() =  comp 201
```

# strncat( )

- Concatenates (appends) portion of one string at the end of another string.
- strncat ( char * destination, char * source, size_t num );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
   char source[20] = " Spring2022" ;
   char target[20]= " comp201" ;
   printf ( "\nSource string = %s", source ) ;
   printf ( "\nTarget string = %s", target ) ;
   strncat ( target, source, 7 ) ;
   printf ( "\nTarget string after strncat() = %s", target ) ;
}
```

# strncat( )

- Concatenates (appends) portion of one string at the end of another string.
- strncat ( char * destination, char * source, size_t num );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
   char source[20] = " Spring2022" ;
   char target[20]= " comp201" ;
   printf ( "\nSource string = %s", source ) ;
   printf ( "\nTarget string = %s", target ) ;
   strncat ( target, source, 7 ) ;
   printf ( "\nTarget string after strncat() = %s", target ) ;
}
```

Output:

```
Source string =  Spring2022
Target string =  comp201
Target string after strncat() =  comp201 Spring
```

# strcpy( )

- Copies contents of one string into another string.
- strcpy ( char * destination, char * source );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
    char source[ ] = "comp201" ;
    char target[20]= "" ;
    printf ( "\nsource string = %s", source ) ;
    printf ( "\ntarget string = %s", target ) ;
    strcpy ( target, source ) ;
    printf ( "\ntarget string after strcpy( ) = %s", target ) ;
    return 0;
}
```

# strcpy( )

- Copies contents of one string into another string.
- strcpy ( char * destination, char * source );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
    char source[ ] = "comp201" ;
    char target[20]= "" ;
    printf ( "\nsource string = %s", source ) ;
    printf ( "\ntarget string = %s", target ) ;
    strcpy ( target, source ) ;
    printf ( "\ntarget string after strcpy( ) = %s", target ) ;
    return 0;
}
```

Output:

```
source string = comp201
target string =
target string after strcpy( ) = comp201
```

# strncpy( )

- Copies portion of contents of one string into another string.
- strncpy ( char * destination, char * source, size_t num );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
    char source[ ] = "comp201" ;
    char target[20]= "" ;
    printf ( "\nsource string = %s", source ) ;
    printf ( "\ntarget string = %s", target ) ;
    strncpy ( target, source, 4 ) ;
    printf ( "\ntarget string after strncpy( ) = %s", target ) ;
    return 0;
}
```

# strncpy( )

- Copies portion of contents of one string into another string.
- strncpy ( char * destination, char * source, size_t num );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
    char source[ ] = "comp201" ;
    char target[20]= "" ;
    printf ( "\nsource string = %s", source ) ;
    printf ( "\ntarget string = %s", target ) ;
    strncpy ( target, source, 4 ) ;
    printf ( "\ntarget string after strncpy( ) = %s", target ) ;
    return 0;
}
```

Output:

```
source string = comp201
target string =
target string after strncpy( ) = comp
```

# strlen( )

- Gives the length of the given string.
- strlen (char * str );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
    int len;
    char array[20]="comp201" ;
    len = strlen(array) ;
    printf ( "\nstring length  = %d \n" , len ) ;
    return 0;
}
```

# strlen( )

- Gives the length of the given string.
- strlen (char * str );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
    int len;
    char array[20]="comp201" ;
    len = strlen(array) ;
    printf ( "\nstring length  = %d \n" , len ) ;
    return 0;
}
```

Output:

```
string length  = 7
```

# strcmp( )

- Compares two given strings and returns zero if they are same.
- If length of string1 < string2, it returns < 0 value.
- If length of string1 > string2, it returns > 0 value.
- strcmp (char * str1, char * str2 );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
   char str1[ ] = "comp" ;
   char str2[ ] = "comp201" ;
   int i, j, k ;
   i = strcmp ( str1, "comp" ) ;
   j = strcmp ( str1, str2 ) ;
   k = strcmp ( str1, "c" ) ;
   printf ( "\n%d \n%d \n%d", i, j, k ) ;
   return 0;
}
```

# strcmp( )

- Compares two given strings and returns zero if they are same.
- If length of string1 < string2, it returns < 0 value.
- If length of string1 > string2, it returns > 0 value.
- strcmp (char * str1, char * str2 );

```c
#include <stdio.h>
#include <string.h>
int main( )
{
   char str1[ ] = "comp" ;
   char str2[ ] = "comp201" ;
   int i, j, k ;
   i = strcmp ( str1, "comp" ) ;
   j = strcmp ( str1, str2 ) ;
   k = strcmp ( str1, "c" ) ;
   printf ( "\n%d \n%d \n%d", i, j, k ) ;
   return 0;
}
```

Output:

```
0
-50
111
```

# strchr( )

- Returns pointer to the first occurrence of the character in a given string.
- strchr(char *str, character);

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a string for testing";
  char *p;
  p = strchr (string,'i');
  printf ("Character i is found at position %lu\n",p-string+1);
  printf ("First occurrence of character \"i\" in \"%s\" is"" \"%s\"",string, p);
   return 0;
}
```

# strchr( )

- Returns pointer to the first occurrence of the character in a given string.
- strchr(char *str, character);

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a string for testing";
  char *p;
  p = strchr (string,'i');
  printf ("Character i is found at position %lu\n",p-string+1);
  printf ("First occurrence of character \"i\" in \"%s\" is"" \"%s\"",string, p);
   return 0;
}
```

Output:

```
Character i is found at position 3
First occurrence of character "i" in "This is a string for testing" is "is is a string for testing"
```

# strrchr( )

- Returns pointer to the last occurrence of the character in a given string.
- strrchr(char *str, character);

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a string for testing";
  char *p;
  p = strrchr (string,'i');
  printf ("Character i is found at position %lu\n",p-string+1);
  printf ("Last occurrence of character \"i\" in \"%s\" is" \
          " \"%s\"",string, p);
   return 0;
}
```

# strrchr( )

- Returns pointer to the last occurrence of the character in a given string.
- strrchr(char *str, character);

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a string for testing";
  char *p;
  p = strrchr (string,'i');
  printf ("Character i is found at position %lu\n",p-string+1);
  printf ("Last occurrence of character \"i\" in \"%s\" is" \
          " \"%s\"",string, p);
   return 0;
}
```

Output:

```
Character i is found at position 26
Last occurrence of character "i" in "This is a string for testing" is "ing"
```

# strstr( )

- Returns pointer to the first occurrence of the string in a given string.
- strstr(char *str1, char *str2);

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a test string for testing";
  char *p;
  p = strstr (string,"test");
  if(p)
  {
    printf("string found\n" );
    printf ("First occurrence of string \"test\" in \"%s\" is"\
            " \"%s\"",string, p);
  }
  else printf("string not found\n" );
   return 0;
```

# strstr( )

- Returns pointer to the first occurrence of the string in a given string.
- strstr(char *str1, char *str2);

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a test string for testing";
  char *p;
  p = strstr (string,"test");
  if(p)
  {
    printf("string found\n" );
    printf ("First occurrence of string \"test\" in \"%s\" is"\
            " \"%s\"",string, p);
  }
  else printf("string not found\n" );
   return 0;
```

Output:

```
string found
First occurrence of string "test" in "This is a test string for testing" is "test string for testing"
```

# strtok( )

- Tokenizes/parses the given string using delimiter.
- strtok ( char * str, char * delimiters );

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[50] ="Test,string1,Test,string2:Test:string3";
  char *p;
  printf ("String  \"%s\" is split into tokens:\n",string);
  p = strtok (string,",:");
  while (p!= NULL)
  {
    printf ("%s\n",p);
    p = strtok (NULL, ",:");
  }
  return 0;
}
```

# strtok( )

- Tokenizes/parses the given string using delimiter.
- strtok ( char * str, char * delimiters );

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[50] ="Test,string1,Test,string2:Test:string3";
  char *p;
  printf ("String  \"%s\" is split into tokens:\n",string);
  p = strtok (string,",:");
  while (p!= NULL)
  {
    printf ("%s\n",p);
    p = strtok (NULL, ",:");
  }
  return 0;
}
```

Output:

```
String is split into tokens:
Test
string1
Test
string2
Test
string3
```

KOÇ UNIVERSITY