Introduction to Programming

Week – *10*, Lecture – *1*

Debugging with gdb

SAURABH SRIVASTAVA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT KANPUR

What is debugging?

The problems in a program or a software are also called *bugs*

- Thus, the process of finding and removing them is called *debugging*

What is debugging?

The problems in a program or a software are also called *bugs*

- Thus, the process of finding and removing them is called *debugging*

We perform debugging usually in two cases

- When the program is running without any runtime errors, but not producing expected results
- When the program is encountering some runtime error, producing a Segmentation Fault

What is debugging?

The problems in a program or a software are also called *bugs*

- Thus, the process of finding and removing them is called *debugging*

We perform debugging usually in two cases

- When the program is running without any runtime errors, but not producing expected results
- When the program is encountering some runtime error, producing a Segmentation Fault

The most commonly used debugger in the C/C++ domain is gdb

What is debugging?

The problems in a program or a software are also called *bugs*

- Thus, the process of finding and removing them is called *debugging*

We perform debugging usually in two cases

- When the program is running without any runtime errors, but not producing expected results
- When the program is encountering some runtime error, producing a Segmentation Fault

The most commonly used debugger in the C/C++ domain is gdb

However, not every program is “debug-able”

- The program’s executable needs to encapsulate some information which gdb needs

What is debugging?

The problems in a program or a software are also called *bugs*

- Thus, the process of finding and removing them is called *debugging*

We perform debugging usually in two cases

- When the program is running without any runtime errors, but not producing expected results
- When the program is encountering some runtime error, producing a Segmentation Fault

The most commonly used debugger in the C/C++ domain is gdb

However, not every program is “debug-able”

- The program’s executable needs to encapsulate some information which gdb needs
- This information must be included in the executable, when you compile and link programs

What is debugging?

The problems in a program or a software are also called *bugs*

- Thus, the process of finding and removing them is called *debugging*

We perform debugging usually in two cases

- When the program is running without any runtime errors, but not producing expected results
- When the program is encountering some runtime error, producing a Segmentation Fault

The most commonly used debugger in the C/C++ domain is gdb

However, not every program is “debug-able”

- The program’s executable needs to encapsulate some information which gdb needs
- This information must be included in the executable, when you compile and link programs
- To do so while using gcc (or g++), you must add the –g switch, e.g.

```
gcc -g -o executable program.c
```

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

Run the executable

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

Run the executable

While the program is executing

- Use `n` to go execute the current line and go to the next line

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

Run the executable

While the program is executing

- Use `n` to go execute the current line and go to the next line
- Use `s` to go inside the function at the current line

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

Run the executable

While the program is executing

- Use `n` to go execute the current line and go to the next line
- Use `s` to go inside the function at the current line
- Use `c` to continue execution till the next breakpoint is hit (or a watched expression's value changes)

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

Run the executable

While the program is executing

- Use `n` to go execute the current line and go to the next line
- Use `s` to go inside the function at the current line
- Use `c` to continue execution till the next breakpoint is hit (or a watched expression's value changes)

Use the `p` command at any point to print values of variables and expressions

The debugging process

Open the executable with gdb, e.g.

```
gdb ./executable
```

Set breakpoints and watchpoints

- This can be done after the program starts executing as well

Run the executable

While the program is executing

- Use `n` to go execute the current line and go to the next line
- Use `s` to go inside the function at the current line
- Use `c` to continue execution till the next breakpoint is hit (or a watched expression's value changes)

Use the `p` command at any point to print values of variables and expressions

If you are revising right now, watch the lecture now for a live demonstration