Lab 3

School of Architecture, Civil and Environmental Engineering

EPFL, SS 2019-2020

http://disal.epfl.ch/teaching/signals_instruments_systems/

Lab 3 outline

- This lab has three main goals:
  - Pointers
  - Structures in C
  - How to debug
- New tools:
  - GDB debugger

Pointers

- A pointer is a **variable** that contains **the address of another variable**.
- A pointer can be declared as follows:

```
Type* name
```

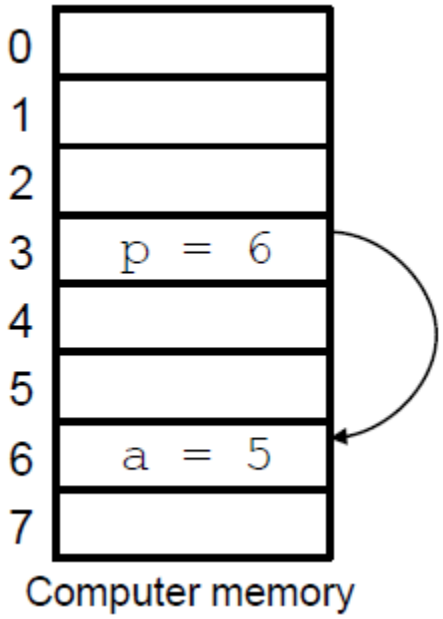
- A pointer can be dereferenced using symbol *

```
(*name)
```

- To obtain the address of another variable, the operator & can be used:

```
int a = 5;
int* p = &a;
```

	Type	Address	Value
a	int	6	5
p	int*	3	6



Pointers

- Never access the memory you did not allocate (e.g., exceed an array bound)

- Always use **`malloc`** and **`free`** in pair

```
Type* pName =(Type*)malloc(length*sizeof(Type));
free(pName);
```

- Be careful about termination character (`\0`) in strings

struct basics

- Definition of a structure:

```
struct <struct-type>{
 <type> <identifier>;
 <type> <identifier>;
 ...
} ;
```

Each identifier defines a member of the structure.

- Example:

```
struct Date {
 int day;
 int month;
 int year;
} ;
```

The “Date” structure has 3 members, day, month & year.

Using struct

```
struct Date {
    int day;
    int month;
    int year;
} ;
```

- `struct Date date1;`
  - instance
- `struct Date* date2;`
  - Pointer to a struct
- For accessing day:
  - `date1.day`
  - `date2->day`

What is gdb?

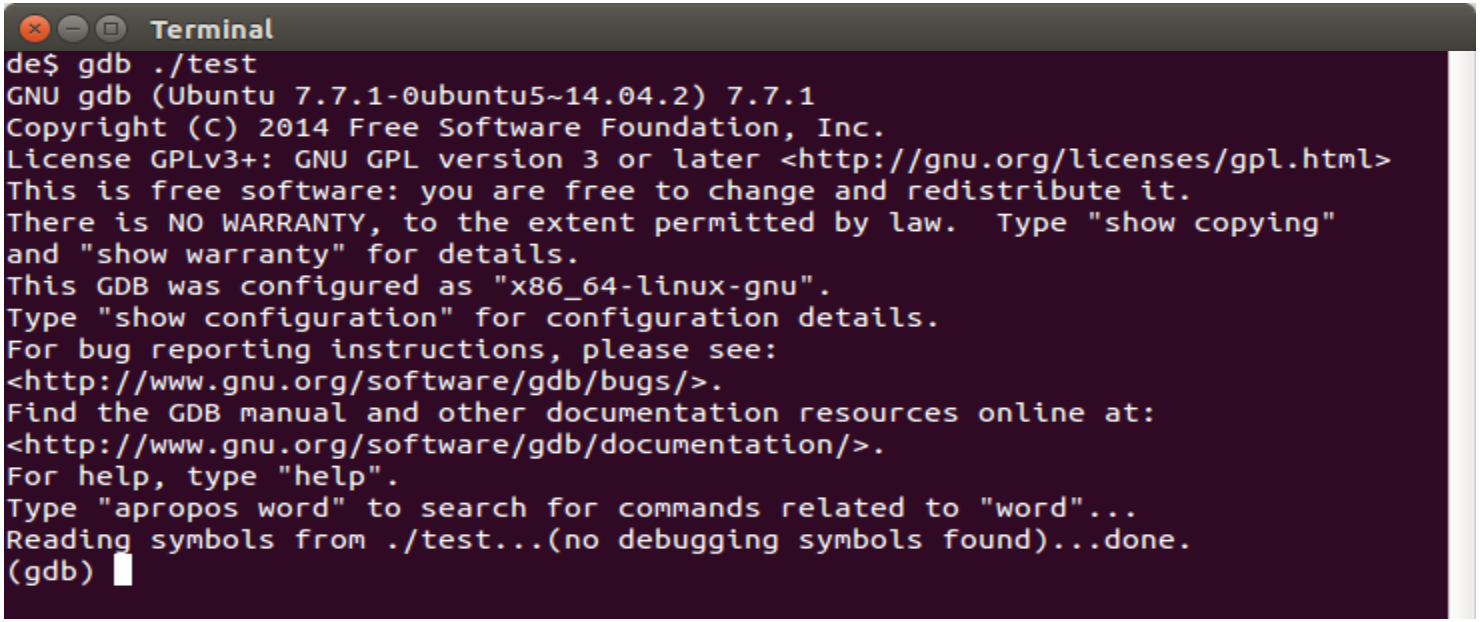
- GDB is the GNU Project debugger
- gdb provides some helpful functionality
  - Allows you to stop your program at any given point.
  - You can examine the state of your program when it's stopped.

Using gdb:

- To start gdb with your hello program type:

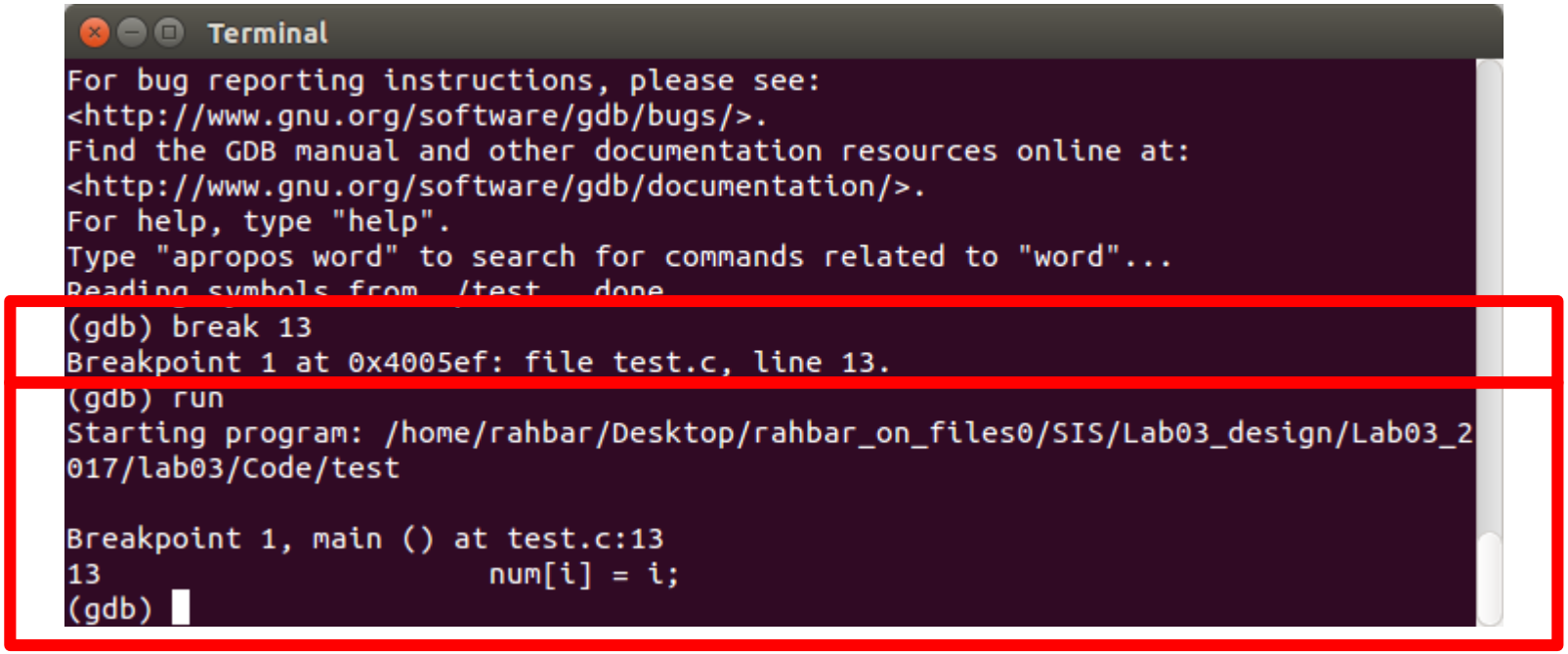
  `gdb ./test`

- When gdb starts, your program is not actually running.



Using gdb:

- You have to use the **`run`** command to start execution.
- Before you do that, you should place some break points.
- Once you hit a break point, you can examine any variable.



Useful gdb commands

- **`break` *`place`***
  - ***`place`*** can be the name of a function or a line number
  - For example: **`break main`** will stop execution at the first instruction of your program
- **`delete` *`N`***
  - Removes breakpoints, where ***`N`*** is the number of the breakpoint
- **`step`**
  - Executes current instruction and stops on the next one
- **`next`**
  - Same as **`step`** except this doesn't step into functions

Gdb commands cont.

- **`run` *`command-line-arguments`***
  - Begin execution of your program with arguments
- **`print` *`E`***
  - Prints the value of any variable in your program when you are at a breakpoint, where ***`E`*** is the name of the variable you want to print
- **`help` *`command`***
  - Gives you more information about any command or all if you leave out command
- **`quit`**
  - Exit gdb

Feedback for Lab 3

Please help us improve the labs by giving us feedback.

Thank you!