CS2100: Computer Organisation

Lab #2: Debugging using GDB II

(Week 4: 4 – 8 September 2023)

[This document is available on Canvas and course website https://www.comp.nus.edu.sg/~cs2100]

Name: _____ Student No.: _____

Lab Group: _____

Special Note for Users Using MacOS on Apple Silicon

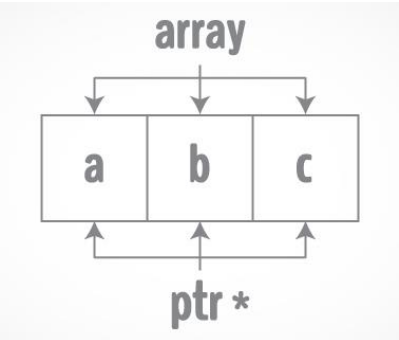
The GDB debugger is unfortunately still unavailable for users of MacOS on Apple Silicon (M1/M2 based MacBooks, for example). If you are using MacOS on Apple Silicon, there are two main choices for you:

i) Purchase and install Parallels, then install Ubuntu. GDB works on Ubuntu running on Apple Silicon. Parallels is expensive, though there is a student discount available at https://www.parallels.com/landingpage/pd/education/. An advantage of Parallels is that you can run MacOS, Ubuntu and Windows applications side-by-side without having to reboot.

ii) Use LLDB instead of GDB. The commands to achieve each step may be different and you will have to work harder on this lab, but it is a viable option for you to learn how to debug C programs on MacOS running on Apple Silicon. You can find an LLDB tutorial here: https://lldb.llvm.org/use/tutorial.html

C Arrays

Array is a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

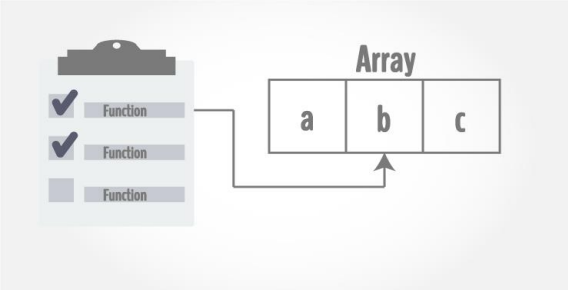


Instead of declaring individual variables, such as number0, number1... and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index which starts from 0.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

C Functions and Arrays

In C programming, a single array element or an entire array can be passed to a function. A single value will be passed by value, whereas when passing the whole array, it is always passed as a pointer to the first element of the array.



Objective:
You will learn how to use arrays and functions in C.

Preparation (before the lab):
Please refer to lab#1 if you have not yet installed gdb on your system.

Procedure:

1. Retrieve the **lab2a.c** and **lab2b.c** programs.

2. Compile **lab2a.c** with **gcc** using the following command: **`gcc -o lab2a lab2a.c`**

3. What is the output of the program? Can you change it to **"2"**?
   Note: The output should be related to the **`ageArray`** such as an element in **`ageArray`**.

4. What is the purpose of the operator **`sizeof`**? What datatype will **`sizeof`** always give **"1"** value for on all architectures?

5. Can you get the number of elements in **`ageArray`**? To produce the following output:

```
2
Size of the array is 4
```

Modify the main function, write it below and show the output to your labTA.
Note: The output **"2"** and size of array (i.e., **4** (*four*)) should be related to **`ageArray`** such as an element in **`ageArray`** and the number of elements in **`ageArray`**.

6. Compile **lab2b.c** with **gcc** using the following command: **`gcc –o lab2b lab2b.c`**

7. Can you give 2 ways of displaying the stored value and address value of the first element of an array?

8. Can you define the function **`hexToDecimal(char hex[], size_t size)`** in the lab2b.c using pointers to traverse the array? Write your function below and show your labTA the output.
Note: You are not allowed to use **`strtoul`**, **`strtol`**, or other functions from **`stdlib.h`**.
Hint: Reading the hexadecimal numbers backwards might be easier. Furthermore, you are already given the function ***`hexVal(char hex)`*** *to simplify your work.*

9. Why do we pass the size of the array to the **`hexToDecimal`** function in lab2b.c? Can we calculate the size of the array inside the function?

10. What is the format specifier to print a variable of datatype **`size_t`**?

Marking Scheme: Report – 5 marks; correct output – 5 marks; Total: 10 marks.

Program lab2a.c

```
#include <stdio.h>

void display(int);

int main() {
    int ageArray[] = { 2, 15, 4, 23 };
    display(ageArray[2]);
    return 0;
}

void display(int age) {
    printf("%d\n", age);
}
```

Program lab2b.c

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

int hexToDecimal(char[], size_t);
int hexVal(char);
```

```
int main(void) {
	// As a basic requirement, translate just the first two-digit
	// hex number. As an extra exercise translate all digits.
	char hex[10];
	size_t len;

	printf("Enter up to 8 hexadecimal digits (e.g. 091A2C, etc): ");
	fgets(hex, 10, stdin);
	len = strlen(hex);

	/* End-of-Line Check */
	if(hex[len-1] == '\n') {
		len = len - 1;
		hex[len] = '\0';
	}
	printf("You entered: %s\n", hex);
	printf("The value in decimal is: %d\n", hexToDecimal(hex, len));

	return 0;
}

int hexVal(char hex) {
	switch(toupper(hex)) {
		case '0': return 0;
		case '1': return 1;
		case '2': return 2;
		case '3': return 3;
		case '4': return 4;
		case '5': return 5;
		case '6': return 6;
		case '7': return 7;
		case '8': return 8;
		case '9': return 9;
		case 'A': return 10;
		case 'B': return 11;
		case 'C': return 12;
		case 'D': return 13;
		case 'E': return 14;
		case 'F': return 15;
	}
	return 0;
}

int hexToDecimal(char hex[], size_t size) {
	// complete the function body

	return 0;
}
```