CS 240 Programming in C

Functions

February 11, 2019

Schedule

1. Decimal Digits of Precision
2. Variable-Length Arrays (C99)
3. Intro to GDB
4. Function

Before Class

- It is all right for me to answer repetitive questions.
- And I am encouraging you to ask them to me or our TA who will be assigned later to get things resolved than leave it there.
- But also I encourage you all to pay more attention in class, to blackboard announcements as well as instruction notes.

Before Class

- If you find something confusing about any instruction notes, it is better for you to post it on blackboard discussion section, so that we could fix it for all.

For New Starters

- Most likely, you will stuck with linux command line.
- In this situation, you could just google "linux command line Basic". There are tons of video and document tutorial.
- Just spend 10-20 minutes to read or watch one or two articles or video, you will get much help than waiting there.
- It may not solve your problem directly, but it is helpful.

For New Starters

- Nonetheless, here are two links:
- `https://www.cs.umb.edu/~ghoffman/linux/linux_help.html`
- `https://www.youtube.com/watch?v=cBokz0LTizk`

Schedule

- Today I will clarify and add something to previous topics.
- Then get into new stuff.

Do-While Loop

First, I want show an example of a do-while loop.
Do-while loop will at least execute the while body once.

```
do{
      statements
} while (expression);
```

Do-While Loop

```
char s[] = "hello";

do{
     printf("%c", s[i++]);
} while (s[i]);
```

Decimal Digits of Precision

- Recall that float type in c has a decimal digits of precision of 6.
- This number means that it is guaranteed to be correct for float type addition within the rounded up highest 6 significant decimal digits.
- This also means most likely it is not correct beyond the 6 decimal digits range.

Decimal Digits of Precision

What will be printed out?

```
printf("1e40: %f\n",1e40 );
printf("2e50: %f\n",2e50 );

printf("1e40 + 2e50: %f\n", 1e40 + 2e50 );
printf("%g\n", 1e40 + 2e50 );
printf("%g\n", 1e40 + 2e46 );
printf("%g\n", 1e40 + 2e45 );
printf("%f\n", 1e40 + 2e45 );

printf("%g\n", pow(2,-126));
```

Floats to Integer

We can get the integer part of a floating point number by foreful typecast like this below:

```
int num = (int) 3.14f;
int num = (int) 3e10f;
```

Floats to Integer

- However, a float type range is much larger than an int type, it is no meaning and undefined behaviour for getting the integral part of a much large float, like this

```
int num = (int) 3e120f;
```

Variable-Length Arrays (C99)

- Recalled that to define an array the length must be a constant integer value and the subscripting could be an integer expression.
- In C99, however, it's possible to use an integer expression to define an array length.
- This practice is not encouraged, since it dynamically allocate memory in stack.
- In this situation, programmers mostly use pointers and malloc() for dynamic memory(heap) allocation.
- `https://en.wikipedia.org/wiki/Variable-length_array`

Example

```
void print(int n){
     int a[n];
     for (int i=0;i<n;i++)
         printf("%d\n", a[i]=i);
}
```

Intro to GDB

- As we are going to write some more lines of code for homework, it is a good start for us to know basic gdb debugging tools.
- Especially for C new starters, there are many subtle pitfalls could cause much trouble and hard to find in a slightly complex code.
- gdb is very helpful for locating where C program goes wrong, like infinite loops and stack overflow etc.

Example 1

Explain what this code does and what is special about it?

```
int i=0,j=0, temp;
int arr[30];
for (i=0;i<30;i++){
    arr[i] = i;
}
while (j < i){
  temp = arr[j];
  arr[j] = arr[i];
  arr[i] = temp;
}
```

Example 1

This program has an infinite loop.

Example 2

How about this one?

```
int a, b[10],i=0;
while (a=9)
    b[i++] = 0;
```

Example 2

This program has an infinite loop and an array subscripting that will go out of range endlessly.

gdb

- The above two programs are of course simple and easy to debug.
- But when these kind of mistakes are embedded in a much large code base, then it is very hard to locate where goes wrong by just reading the code.

gdb

- Compile for debugging:
  gcc -g program.c
- The flag '-g' preservers all the identifiers symbols of variables and functions in program which are lost when the program gets compiled, such that we can utilize when we are debugging.
- Start gdb with program with
  gdb [program]
- Let's do a demo and a couple of commands.

gdb

- `http://users.ece.utexas.edu/~adnan/gdb-refcard.pdf`
- For an infinite loop that runs forever, you just type ctrl + c in gdb and it will end within the loop.
- For segfault it will stop at the line of code where causes segfault.
- There are more to gdb, we will cover later.

Called by Value

- In C, all function arguments are "passed by value".
- This is different from "call by reference" language like Fortran or with var parameters in Pascal, in which the called function has access to the original argument, not a local copy.

Example

What will be print out?

```
int add(int a, int b){
    printf("function a : %p", &a);
    return a+b;
}

int main(void){
    int a=0,b=1;
    printf("main a : %p", &a);
    add(a,b);
    return 0;
}
```

Called by Value

- In C, all function arguments are "passed by value".
- However, we should be careful if the argument is an array variable?

```
Question:

For an array argument, what value has been
passed into function body?

Are all the elements of the array here?
```

Example2

What will be print out?

```
int add_two(int c[]){
    printf("function   c_value : %p\n", c);
    printf("function c_address : %p\n", &c);
}

int main(void){
    printf("      main c_value : %p\n", &c);
    add_two(c);
    printf("    main c_address : %p\n", &c);
    return 0;
}
```

Return by Value

- The same thing goes with "return by value".

```
    Question:
Can an array being returned?
```

Return by Value

- You can return an array, but the return type has to be a pointer.
- The return value cannot be assigned to another array variable. It can only be assigned to a pointer variable.