COMP1521 Tutorial 01

Introduction

- Who am I?
- Who are you!
  - Your name (Great movie)
  - What you study and what year you are in
  - What is your favourite pokemon?

What did we learn in COMP1511/1917/1911?

- Binary
- Memory
- Pointers
- Linked Lists
- Stacks
- Queues
- File I/O
- Assembly

What is COMP1521?

- Bridging the gap between hardware and software
- Assembly
- System Calls and OS interactions

Consider the following C program

```c
#include <stdio.h>

int main(void)
{
    int n = 1234;
    int *p;

    p = &n;
    n++;
    printf("%d\n", *p);
    printf("%p\n", p);
    p++;
    printf("%p\n", p);
    return 0;
}
```

If we assume that the variable `n` has address `0x7654`, then what values will the program print?

Consider the following C program

What is the output from the following program and how does it work? Try to work out the output *without* copy-paste-compile-execute.

```c
#include <stdio.h>

int main(void)
{
    char *str = "abc123\n";
    char *c;

    for (c = str; *c != '\0'; c++)
       putchar(*c);

    return 0;
}
```

Consider the following C program

Consider the following `struct` definition defining a type for points in a three-dimensional space:

```
1. typedef struct _Coord {
2.    unsigned int x;
3.    unsigned int y;
4.    unsigned int z;
5. } Coord;
```

and the program fragment using `Coord` variables and pointers to them

```
1. {
2.    Coord a, b, coords[10], *p;
3.    a.x = 5;  a.y = 6;  a.z = 7;
4.    p = &a;
5.    b.x = 3;  b.y = 3;  b.z = 3;
6. /*A*/
7.    (*p).x = 6;  p->y++;  p->z++;
8.    b = *p;
9. /*B*/
10. }
```

a. Draw diagrams to show the state of the variables `a`, `b` and `p` at points `/*A*/` and `/*B*/`

b. Why would a statement like `*p.x++;` be incorrect?

c. Write code to iterate over the `coords` array using just the variable `p` and setting each item in the array to (0,0,0). Do not use an index variable.

Static Declarations

- ***static*** does different things depending on where it's declared
  - Static variable in function = keeps values between invocations
    - Allocated in *data* segment of memory
    - Always initialized as 0
    - Has to be initiated as a *constant literal*
  - Static function or global variable in file = function\variable only accessible in that file
    - Also known as *internal linkage*

Consider this code

What is the effect of each of the `static` declarations in the following program fragment:

```
1. #include <stdio.h>
2.
3. static int x1;
4. ...
5. static int f(int n)
6. {
7.     static int x2 = 0;
8.     ...
9. }
```

Consider this code

What is the difference in meaning between the following pairs (a/b and c/d) of groups of C statements:

a.
```
if (x == 0) {
   printf("zero\n");
}
```

b.
```
if (x == 0)
   printf("zero\n");
```

c.
```
if (x == 0) {
   printf("zero\n");
   printf("after\n");
}
```

d.
```
if (x == 0)
   printf("zero\n");
   printf("after\n");
```

How C deals with errors

- Terminating the program entirely (rare)
- Setting the system global variable *errno*
- Returning a value that indicates an error (e.g. <@>NULL)
- Setting a returning parameter to an error value

- Could use a combination of the above

Gcc flags

- For each of the following commands, describe what kind of output would be produced:
  - $gcc\ -E\ x.c$
    - Only preprocesses the code
    - Outputs pre-processed source code
  - $gcc\ -S\ x.c$
    - Generates assembly but doesn't assemble it
    - Outputs assembler
  - $gcc\ -c\ x.c$
    - Compiles the source files fully but doesn't link
  - $gcc\ x.c$

Fun with Queues

What is the state of the queue after each of these operations?

- insert 5
- insert 7
- insert 3
- insert 4
- remove
- remove
- insert 6
- insert 1
- remove
- insert 9

More Queue Stuff

- Question 11 -> https://cgi.cse.unsw.edu.au/~cs1521/18s2/tutes/week01/index.php

Stuff that will help with the labs

- File I/O Revision!
- Command Line Arguments