COMP1511 18s1

— Lecture 1 —

Numbers In, Numbers Out

Andrew Bennett

<andrew.bennett@unsw.edu.au>

more printf

variables

scanf

Before we begin…

introduce yourself to the person sitting next to you

why did they decide to study **computing**?

Admin

tutorials and laboratories start in week 1
(some of you have already had tutes and labs)

lecture recordings are on WebCMS 3

make sure you have **home computing** set up

Getting Help

post on the course forum

I (or others) will answer so everyone can see

talk to me after the lecture

talk to your tutor

help labs

admin:

cs1511@cse.unsw.edu.au

if it needs to go to me directly:

andrew.bennett@unsw.edu.au

Hello World

```
// Prints out a friendly message.
// Andrew Bennett <andrew.bennett@unsw.edu.au>
// 2018-02-27

#include <stdio.h>

int main (void) {

    // Print out the famous 'hello world' message.
    printf ("Hello, world!\n");

    return 0;
}
```

Navigating on Unix

pwd shows where you currently are

```
$ pwd
/import/ravel/2/andrewb
```

ls lists the items in the current directory

```
$ ls
18s1    bin     lib     public_html     tmp     web
```

mkdir makes a new directory

```
$ mkdir cs1511
$ ls
18s1    bin     cs1511  lib     public_html     tmp     web
```

Navigating on Unix

cd changes directory

```
$ cd cs1511
$ pwd
/import/ravel/2/andrewb/cs1511
$ ls
$
```

cd .. changes into the previous directory

```
$ cd ..
$ pwd
/import/ravel/2/andrewb
```

Writing a Program

to create a C program from the terminal,
open a text editor like **gedit**

```
$ gedit hello.c &
```

once the code is written and saved...
compile it with **dcc**!

```
$ dcc -o hello hello.c
```

Programming is a construction exercise

think about the problem

write down a proposed solution

break each step into smaller steps

convert the basic steps into instructions in the programming language

use an *editor* to create a *file* that contains the program

use the *compiler* to check the *syntax* of the program

test the program on a range of data

Compiling

remember: we write C programs for **humans** to read.

a C program must be translated into *machine code* to be run.

this process is known as *compilation*,
and is performed by a *compiler*.

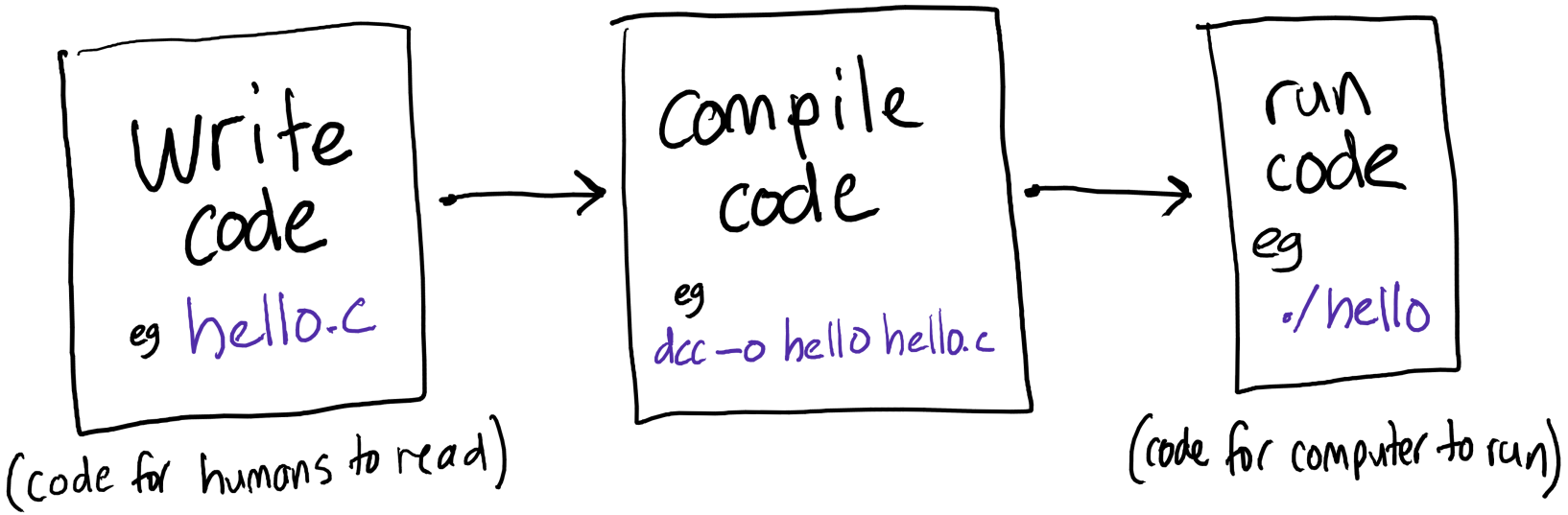
we will use a compiler named **dcc** for COMP1511
dcc is actually a custom wrapper around a compiler named clang.

another widely used compiler is called **gcc**.

The overall process

"What's the difference between the dcc command and the ./ command?"

There are three steps to writing + running code:



(re-)introducing: printf

```
printf ("hello world!\n")
```

prints the text

`"hello world!"`

to the terminal

Printing more than just words

Can we print more than just words?

Printing more than just words

Can we print more than just words?

…

Yes!

`printf`

the *f* stands for “formatted”

Let’s try it out!

```c
// Prints out the sum of two numbers.
// Andrew Bennett <andrew.bennett@unsw.edu.au>
// 2018-02-28

#include <stdio.h>

int main (void) {
    // Sum two numbers, and print them out.
    printf ("The sum of 3 and 5 is %d\n", 3 + 5);

    return 0;
}
```

we can change the numbers
to add different values together
and print them out!

but that’s boring if it can’t be dynamic,
and it sucks to do it by hand.

Introducing **variables!**

Variables and Types

Variables are used to store data…
think “boxes”

Each variable has a **data type**…
this is the size and structure of the “box”

For the next few weeks, we’ll only use two data types:

int stores whole numbers:
`2`, `17`, `-5`

float stores “floating-point” numbers:
`3.14159`, `2.71828`

we’ll look at other types in future weeks. can you think of any other types that would be useful?

Variables

declare

the first time a variable is mentioned,
we need to specify its type.

initialise

before using a variable we need to assign it a value.

assign

to give a variable a value.

```
int num; // Declare
num = 5; // Initialise (also Assign)
...
num = 27; // Assign
```

Variables

we can also declare and initialise in the same step:

```
int num = 5; // Declare and Initialise
...
num = 27; // Assign
```

Variable Naming (and other identifiers)

must be made up of letters, digits and underscores (`_`)

the first character must be a letter

are case sensitive (`num1` and `Num1` are different)

Keywords like

`if`, `while`, `do`, `int`

cannot be used as identifiers

Printing Variables Out

No variables:

```
printf ("Hello World\n");
```

A single variable:

```
int num = 5;
printf ("num is %d\n", num);
```

Printing Variables Out

More than one variable:

```
int num1 = 5;
int num2 = 17;
printf("num1 is %d and num2 is %d\n", num1, num2);
```

The order of arguments
is the order they will appear:

```
int num1 = 5;
int num2 = 17;
printf ("num2 is %d and num1 is %d\n", num2, num1);
```

`printf`'s placeholders

int uses `%d`

Numbers in: `scanf`

```
int num = 0;
scanf ("%d", &num);
printf ("num = %d\n", num);
```

Note that the variable is still initialised.
(Not necessary, but good practice.)

Note the & before the variable name.
Don’t forget it!

Reading Variables In

Multiple variables (space separated):

```
int num1 = 0;
int num2 = 0;
scanf ("%d %d", &num1, &num2);
printf ("num1 = %d and num2 = %d\n", num1, num2);
```

Multiple variables (comma separated):

```
int num1 = 0;
int num2 = 0;
scanf ("%d, %d", &num1, &num2);
printf ("num1 = %d and num2 = %d\n", num1, num2);
```

Note the space or comma between the variables.

making **decisions**

different behaviour in different situations

Driving, Take 1

Write a program which asks the user to enter their age.

If they are at least 16 years old,
then, display “You can drive.”

Then, whether or not they can drive,
display “Have a nice day.”

Driving, Take 1

Write a program which asks the user to enter their age.

If they are at least 16 years old,
then, display “You can drive.”

Then, whether or not they can drive,
display “Have a nice day.”

Driving, Take 1: Step by Step

… Print “How old are you?”

… Read in their age.

… If their age is ≥ 16: print “You can drive”.

… Print “Have a nice day.”

```c
// Can a user drive?
// Andrew Bennett <andrew.bennett@unsw.edu.au>
// 2017-07-31

#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
    printf ("How old are you? ");
    int age = 0;
    if (age >= 16) {
        printf ("You can drive.\n");
    }

    printf ("Have a nice day.\n");

    return EXIT_SUCCESS;
}
```

Detour: Defining Constant Values

Using the same value numerous times in a program
becomes high maintenance if the value changes…
and needs to be changed in many places.
(You may miss one!)

Other developers may not know (or you may forget!)
what this magical number means.

```
#define MIN_DRIVING_AGE 16
```

note

there is no semicolon at the end of this line

Feedback?

bit.do/comp1511-feedback-week1