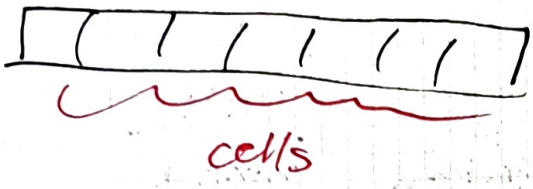
Wk 1 COMP2017 Lecture Note

Memory

* Intro to Memory



cells

Q How much memory do we have?

Q where in the memory is the value we want to retrieve?

Q where does the newly stored value go?

Q How do we refer to an area of the memory?

pointers. "for someone else"

The addressing system

Memory is all about (Similar to arrays (with indices & values)

- addresses : location in memory of a value
- values : arbitary number of bits
  - ↳ every value has an address.

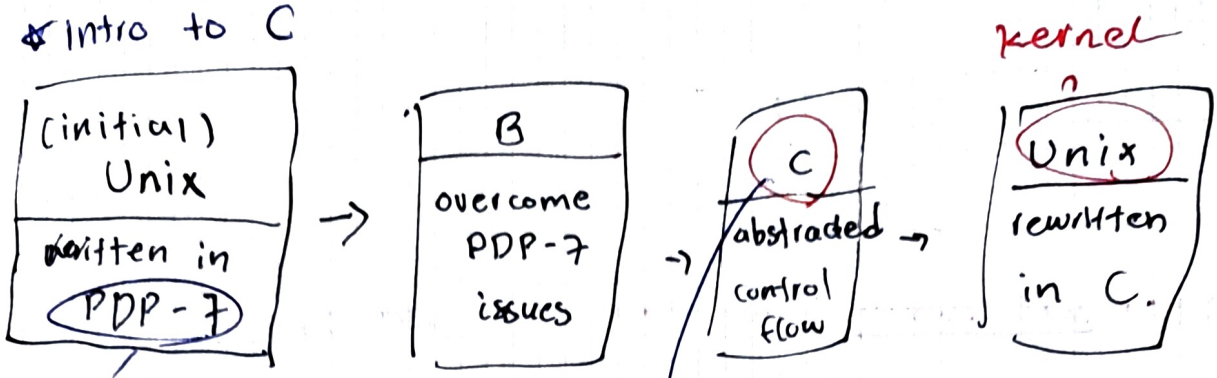
Computers calculate & copy (not move) memory

(

Calculations rely on the memory being organised.

- ↳ correct values were copied
- ↳ correct address was accessed
- ↳ at the right time !!! (addresses can shift)

Intro to C



python, unix kernel, many other programs/projects.

C
- actual C-language
- pre-processing language
  - ↳ input/output of text (commands) ~~#include~~ ~~#define~~ #de
  - ↳ Text-macro language "text-manipulation") declaration for variables & functions in the header
  - ↳ conditional compilation

Uniqueness of C.

- No garbage collection
  - use pointers
  - free & assign memory manually
- Program contain files for storing modules (compiled code)
  - ↳ global var
  - functions (main is the first)
  - local vars in each functions.

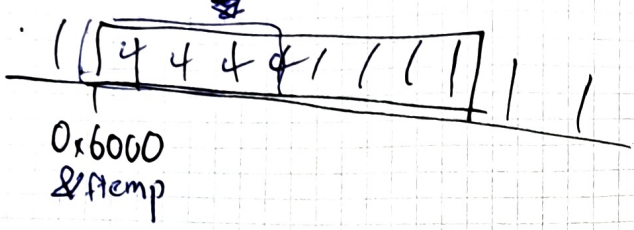
```
#include <>          array of
                     array of chars = str
int main(int argNum, char **argv)
{ printf("Hello");
  return 0;
}
```

```
int ftemp;
```

(&) ampersand = address of

```
scanf("%d", &ftemp)
```

- read in enough bits for signed integer



In a larger project

- Modules : single files
- modules are translated to object files
- object files can be ~~linked~~ ?

referenced var

eg. extern int num1;

extern int foo(int x, int y);

↳ referenced function

int

%d = signed int

%u = unsigned int

%f or %.2f = float with/without specific decimal point

%s = string %ld = long

%p = pointer %x = hexadecimal