CS 537
Section 1
Programming in Unix and C

Michael Swift

Project notes

- First project is individual
  - The rest are for groups
    - How large a group?
    - How should we assign credit?

Facilities

- Department Linux machines (penguins):
  - 1350: emperor
  - 1351: king
  - 1370: adelie, humboldt, macaroni
- Unix Orientation classes
  - Wednesday, Thursday, Monday at 4 pm in room 1325

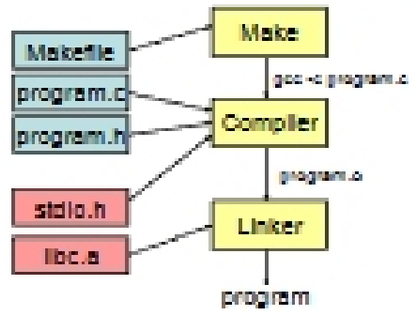
Why C

- All modern operating systems are written in C
- Why?
  - Control
  - Predictable code
  - Expressive
  - Optimizable
  - Powerful pre-processor

Issues with C

- Little hand-holding for programmer
  - Manual memory management
  - Small standard library
  - No native support for threads and concurrency
  - Weak type checking

Using C and Unix



C language

```
#include <stdio.h>
int main(int argc, char * argv[])
{
  printf("Hello, world\n");
  return(0);
}
```

Issues with C

- Memory allocation
  - malloc(), free()
- Pointer arithmetic and arrays
- Preprocessor

Example

Memory

- You have to mange memory yourself.
- Stack allocated memory becomes invalid when you return from function. This will not work:

```
char *f() {
  char str[100];
  strcpy(str,"hello, world\n");
  return(str);
}
```

- Memory from malloc only becomes invalid when you free it:

```
char *f() {
  char *str;
  str = malloc(100);
  strcpy(str,"hello, world\n");
  return(str);
}
```

- Is ok., but someone has to call free(str);

Strings

- Strings in C are arrays of bytes:
  - char str[100];
- Or pointers to memory
  - char * str;
  - str = malloc(100);
- They are null terminated – so you need to make space for it
  - str[2] = '\0';
  - strlen(str) = 0;
- There are a bunch of functions for working with them:
  - strlen, strcpy, strcat

File I/O

- f* functions for accessing files:
  - struct FILE *: represents an open file
  - f = fopen("foo","r") – open file foo for reading
  - fclose(f) – says you are done with f
  - bytes = fread(buffer,size,count,f) = reads size x count bytes from f into buffer
  - fwrite(buffer, size, count ,f) = writes size x count bytes to f from buffer
  - fgets(str, size, f) = reads up to size-1 bytes from a single line of f into str, including newline