Software Tools

C, Unix (Linux), and tools

Register Variables

- Declared with
  - `register int n;`
- Advise the compiler to put them into CPU registers
- Gives a hint for optimization
- Only certain types can go to registers
- One cannot get the address of a register
- The compiler may ignore the advice

Block structure

- C is not a block structured language in the general sense
  - One cannot define functions within functions with complex scope rules
- But we can put block within blocks
  - We can define and initialize variables local to the blocks
  - Such variables “hide” variables with the same name outside the block
  - We can even define them as static

Variable initialization

- Variables can be initialized when defined
  - Static and external variables with constant expressions only. Initialization happens at the begining of execution of the program
  - Automatic variables can be anything that makes sense. Initialization happens every time the function is called.
- Without initialization variables get default values
  - Static and external initialized to zero
  - Automatic initialized to garbage or zero
    - Guaranteed to cause maximum embarassment!

Initialization vs explicit assignment

- It is a matter of preference
- Most programmer would initialize at least some static and external variables
- The author of the textbook and the instructor of the course prefer explicit assignments most of the time.

Recursion

- C, like most languages, supports recursion
- Recursion is a powerful concept
  - Many algorithms are inherently recursive
- It is usually less efficient than iteration
- Compilers can do a very good job optimizing recursive functions

Print Decimals recursively

```
Void printd(int n)
{
  if (n<0){
    putchar('-');
    n=-n;
  }
  if (n/10!=0)
    printd(n/10);
  putchar(n%10+'0');
}
```

C preprocessor

- We can include files with
  - `#include <stdio.h>`
  - `#include “rpolish.h”`
- With angle brackets the preprocessor searches the standard directories for the file
- With quotes it searches the current directory first, then the standard ones

Macro substitution

- Anything can be redefined
  - `#define while if`
- We can even have arguments
  - `#define MAX(A,B) ((A)>(B)?(A):(B))`
- Tricky: what about
  - `MAX(i++,j--)`
- Even worse
  - `#define square(x) x*x`
- Disaster:
  - `square(x+1)`

Preprocessor acrobatics

- We can do some fancy things
  - `#define DBGPRINT(x) {printf(#x, "=%d\n",x);}`
- Or
  - `#define PASTE(A, B) A ## B`
- Rarely used

Preprocessor Conditionals

- We have if statements of various forms
  - `#if MACROA == MACROB`
  - `#if defined(MACROC)`
  - `#ifdef MACROD`
  - `#ifndef MACROD`
- And of course matching clauses
  - `#endif`
  - `#elif`

Problems