Class Information

ANNOUNCEMENTS

• Third homework due Friday, October 13, 11:59pm.

• First project has been posted, due Monday October 23, 11:59pm.

• **Midterm exam**: Friday, October 27, in class.

• Don’t forget to work on your C and Linux skills (ilab).

• **Final exam**: Wednesday, December 20, 4:00-7:00pm.
Imperative Programming Languages

Imperative:
Sequence of state-changing actions.

- Manipulate an abstract machine with:
  1. Variables naming memory locations
  2. Arithmetic and logical operations
  3. Reference, evaluate, assign operations
  4. Explicit control flow statements
- Key operations: Assignment and “Goto”
- Fits the von Neumann architecture closely

Von Neumann Architecture

The given bus widths are examples only!
Run-time storage organization

Typical memory layout

Logical Address Space

<table>
<thead>
<tr>
<th>Code</th>
<th>Static</th>
<th>Heap</th>
<th>free memory</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>S</td>
<td>H</td>
<td></td>
<td>S</td>
</tr>
<tr>
<td>o d e</td>
<td>t a t i c</td>
<td>e a p</td>
<td></td>
<td>t a c k</td>
</tr>
</tbody>
</table>

The classical scheme

- allows both stack and heap maximal freedom
- code and static may be separate or intermingled

Will talk about this in more detail in a later lecture!
C: An Imperative Programming Language

Expressions: include procedure and function calls and assignments, and thus can have side-effects

Control Structures:

• if statements, with and without else clauses

• loops, with break and continue exits

  while ( <expr> ) <stmt>
  do <stmt> while ( <expr> )
  for ( <expr> ; <expr> ; <expr> ) <stmt>

• switch statements

• goto with labelled branch targets
C Examples

```c
while ( ( c = getchar() ) != EOF ) putchar(c);

for ( i = 0 ; s[i] == ' ' ; i++ );

for ( i = 0 ; i < n ; i++ ) {
  if ( a[i] < 0 ) continue; /*skip neg elems*/
  ....
}

c = getchar();
switch(c) {
  case '0': case '1': case '2': case '3':
  case '4': case '5': case '6': case '7':
  case '8': case '9':
    digit[c-'0']++;
    break;
  case ' ': case '
': case 't':
    delim++;
    break;
  ...
}
```
Data Types in C

- Primitives: `char`, `int`, `float`, `double
  no Boolean—any nonzero value is true

- Aggregates: arrays, structures

```c
char a[10], b[2][10];

struct rectangle {
    struct point p1;
    struct point p2;
}
```

- Enumerations: collection of sequenced values

- Pointers:
  &i  address of i
  *p  dereferenced value of p
  p+1  pointer arithmetic

```c
int *p, i;
p = &i;
*p = *p + 1;
```
Next Lecture

Things to do:
Keep working on the project!

Read Scott: Chap. 3.1 - 3.4; 8.3; ALSU Chap. 7.1 - 7.3

Next time:

• More on C and pointers
• Dynamic vs. static scoping
• Runtime environment
• access links and control links management