# ECE 220
# Honors Lab Section

Lab4: Runtime stack & C Miscellanea

# Checkout Script

- Last time you'll ever have to checkout an MP/Lab!!!
- Checkout the 'scripts' directory from SVN
- Demo

# Shells and Standard Streams

- Command shells: command line interface to the OS
  - sh | bash | zsh | fish
- Terminal emulator: application for accessing shell
- 3 standard streams:
  - stdin (0) – data stream going into program
  - stdout (1) – file stream where program writes its output
  - stderr  (2) – file stream where program writes its error

# Piping

- Bash
  - File piping
    - echo “Hello Ouput”
    - echo “Hello Error” 2>
    - echo “Hello File” > random_file.txt
    - echo “Hello Append” >> random_file2.txt
    - echo “Hello Error File” 2> random_error_file.txt
    - echo “Hello Error in Output” 2>&1
    - echo “Hello All” &> random_all.txt (BASH ONLY)
  - Program piping
    - Grading scripts use this
    - Example:
      - ls -l | grep "\.sh$" | nl | cut -c6-

# Regex

- Regular expressions: pattern location and matching

| Symbol | Meaning | Examples |
|---|---|---|
| ^ | Anchor, match at the start of line | ^the = *there* but not brothel |
| $ | Anchor, match at the end of line | ve$ = *have* but not *heaven* |
| […] | Match any character in bracket list | [XYZ] |
| [a-b] | Match range of characters | [a-zA-z0-9] |
| [^…] | Match any character not in bracket list | [^XYZ] |
| . | Matches any single character | ab. = abc, ab_, abz |
| ! | Do not match the next character or expression | ![0-9] |
| ? | Match 0 or 1 time | [01]? |
| * | Match 0 or more times | [01]* |
| + | Match 1 or more times | [01]+ |

# Regex

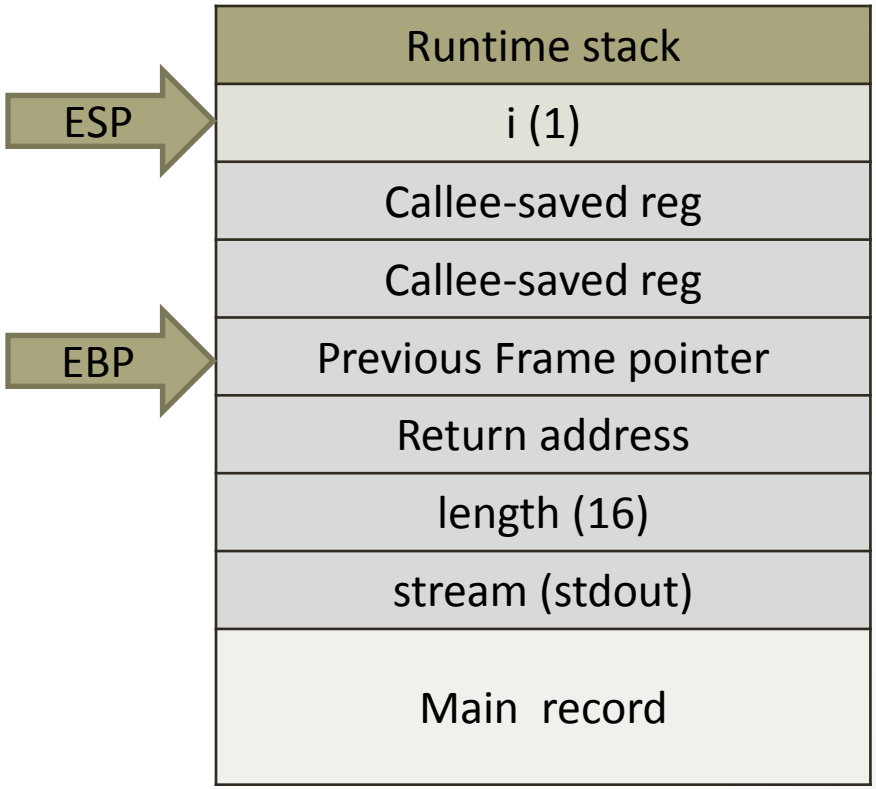
- More

| Symbol | Meaning | Examples |
|---|---|---|
| {n} | Match exactly *n* times | [01]{3} |
| {n,} | Match *n* or more times | [01]{3,} |
| {,n} | Match *n* or fewer times | [01]{,5} |
| {n,m} | Match at least *n* times but no more than *m* times | [01]{3,5} |
| \d (\D) | Matches any single digits | \d = [0-9] |
| \s (\S) | Matches whitespace characters | \scat\s |
| \w (\W) | Matches word | \w=[a-zA-Z0-9_] |
| \b (\B) | Word boundaries | \bcat\b matches cat but not catfish |
| (…) | Determine order of evaluation | (Mon\|Tues)day |
| \ | Escapes a character | \d, \s, \w |

- Example

# Runtime Stack

```
void fizz_buzz(int length, FILE *stream) {
  for (int i = 1; i <= length; i++) {
    if (i % 15 == 0) {
      fprintf(stream, "FizzBuzz ");
    } else if (i % 3 == 0) {
      fprintf(stream, "Fizz ");
    } else if (i % 5 == 0) {
      fprintf(stream, "Buzz ");
    } else {
      fprintf(stream, "%d ", i);
    }
  }
  fprintf(stream, "\n");
}
```



# Runtime Stack Caller x86

- Calling function
  - pushl <caller-regs>
  - pushl <args> #Note this is reverse order
  - call <function>

- Exiting function
  - popl <args>

# Runtime Stack Callee x86

- Entering function
  - pushl %ebp
  - movl %esp, %ebp
  - pushl <callee-regs>
  - subl 12, %esp #Local variables
- Exiting function
  - addl 12 %esp
  - popl <callee-regs>
  - leave
    - movl %ebp, %esp
    - pop %ebp
  - ret
- Example from previous lab

# Arrays

- Initialization
  - type name[length] = {0} or $\{v_1, v_2\}$ or $\{v_1, v_2, \dots, v_{length}\}$
    - double double_arr[2]; What does this initialize to?
    - char char_arr[10] = {0};
    - int int_arr[5] = {1, 2, 3};
    - float float_arr[3] = {1, 2, 3};
  - Elements indexed by 0 to length-1 for length $\geq 1$
- Usage
  - Array notation
    - int temp = int_arr[1]; //Accesses second element of array
  - Pointer notation
    - int *int_ptr = int_arr;
    - int temp = *(int_ptr + 1); //Accesses second element of array