CSCI 2132
Software Development

Lecture 10:

Shells and Computing Environment

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Previous Lecture

- Formatted Input and Output
  - printf function
  - scanf function

scanf Example: More Input Examples

- Consider the same code:

```
int i, j;
double x, y;
scanf("%d%d%lf%lf", &i, &j, &x, &y);
```

- and the following input:

```
1 -20.3 -4.5 5.5
```

- or consider input:

```
1.1 -20 -4.5 .5
```

More examples

- Consider:

```
int i; double x;
scanf("%d %f", &i, &x);
scanf("%d%f", &i, &x);
```

- Are these equivalent? How about these:

```
scanf("%d ", &i);
scanf("%d", &i);
```

- Consider the following:

```
double x, y;
scanf("%f,%f", &x, &y);
scanf("%f ,%f", &x, &y);
```

Example: Adding Fractions

- Program prints: ‘`Enter expression: `’
- Input provided in the form: `2/3+1/6`
- Output: result but not reduced to the lowest terms:

```
15/18
```

- A solution:

```c
#include <stdio.h>

int main() {
    int num1, denom1, num2, denom2,
        result_num, result_denom;

    printf(" Enter expression: ");
    scanf("%d / %d + %d / %d", &num1, &denom1,
          &num2, &denom2);

    result_num = num1 * denom2 + num2 * denom1;
    result_denom = denom1 * denom2;

    printf("%d/%d\n", result_num, result_denom);
    return 0;
}
```

Shells

- Reading: Unix book Chapter 4
- Separated program from kernel
  - not necessarily with all operating systems
- Advantages:
  - Crashing shell does not crash the system
  - Easy to replace or enhance shell without changing the kernel

Shell Functionality

- Built-in commands
- Scripts
- Variables (local and environment)
- Redirection
- Wildcards (file name substitution)
- Pipes
- Sequences (conditional and unconditional)
- Subshells
- Background processing
- Command substitution

Popular Shells

- Bourne shell (`/bin/sh`): This shell replaced the original Thompson shell (which was the first UNIX shell).
- Korn shell (`/bin/ksh`)
- C shell (`/bin/csh`)
- TC shell (`/bin/tcsh`)
- Bash shell (`/bin/bash`)
- Note: Use `cat /etc/shells` to see shells available
- We will focus on the Bash shell

Bash Shell

- We will use bash shell
  - widely available
  - includes many advanced features of other shells
- Default on bluenose
- Command `chsh` used to change default shell
  - not on bluenose, but can ask help-desk
- Command `finger` shows the default shell of a user
- File of interest: `/etc/passwd`

Commands

- built-in (internal) vs. external commands
- built-in commands are generally faster
- some tasks inherently require built-in commands
- examples
  - internal commands: cd, echo, logout
  - external commands: ls, grep, sort, cut, uniq

Shell Variables

- Shell maintains a set of string-valued variables
- typically divided into environment and local variables
- Metacharacter $ used to expand the value of a variable
- Some built-in variables:
  - `SHELL` stores the pathname of the shell
  - `HOME` stores home directory
  - `PATH` stores list of directories to search for commands
  - `PS1` stores default prompt (there are also `PS2`, `PS3`, and `PS4`) — this is bash-specific

Common Shell Variables

- The following are usually common variables among different shells:
- `SHELL` is the full pathname of the login shell
- `HOME` is the full pathname of the home directory
- `PATH` is the list of directories searched for a command
- `USER` is the username
- `MAIL` is the full pathname to the mailbox
- `TERM` is the type of the terminal

Processes

- When we run a program we create a process
- Program vs. process: related but not the same
  - program is inactive piece of code, in a file on disk
- Process occupies memory space consisting of
  - Code (executable machine code)
  - Data (static data)
  - Heap (used for dynamic allocation)
  - Stack (temporary local data used by subroutines)
- We will learn more details about these concepts in C