Introduction

Tom Kelliher, CS 320

Jan. 26, 2011

1 Administrivia

Announcements

Assignment

Read 1.1–4.

Outline

1. Syllabus.
2. Terminal I/O in C.
3. Accessing command-line arguments.

Coming Up

Structures, pointers, and memory allocation in C

2 Syllabus

3 Terminal I/O in C

1. `printf`

```
#include <stdio.h>

int main()
{
   /* Variable declarations must occur at the _start_ of a block. */
   int sum = 12;
   double min = 0.1, max = 5.5;
   char name[] = "Tom Kelliher";

   printf("I am a constant string.\n");

   printf("The sum is: %d\n", sum);

   printf("Min is: %g.  Max is: %g.\n", min, max);

   printf("Your name is %s.\n", name);

   return 0;
}
```

   Refer to `printf(3C)` on phoenix. (`man -s 3C printf`)

2. `scanf`

```
#include <stdio.h>

int main()
{
   int i, age;
   double weight;
   char name[80];

   printf("Enter your age: ");
   scanf("%d", &age);
   printf("You entered %d.\n", age);
```

```
   printf("Enter sample weight: ");
   scanf("%lg", &weight);
   printf("You entered %g.\n", weight);

   printf("Enter your name: ");
   scanf("%s", name);
   printf("Your name is %s.\n", name);

   /* Eliminate whitespace following previous name. */
   while (getc(stdin) != '\n')
      ;

   printf("Enter your name: ");
   fgets(name, 80, stdin);

   /* Eliminate the newline following name */
   i = 0;
   while (name[i] != '\n')
      i++;

   name[i] = '\0';

   printf("Your name is %s.\n", name);

   return 0;
}
```

Refer to `scanf(3C)`.

4 Command-Line Arguments

1. Command-line arguments in Unix:

   ```
   foo arg1 arg2 arg3
   ```

2. Command-line arguments in Eclipse:

3. Command-line arguments in Eclipse: Open the *Run Dialog* for your build command, select the *Arguments* tab, and enter the arguments. Each argument should be separated by a space.

4. Example:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
   int sum, current, i;

   if (argc <= 1)
   {
      printf("No arguments!\n");
      return 1;
   }

   sum = 0;
   for (i = 1; i < argc; i++)
   {
      current = atoi(argv[i]);
      sum += current;
      printf("Arg %d: %d\n",  i, current);
   }

   printf("\nThe sum is %d.\n", sum);

   return 0;
}
```

5. Another example, showing **highly desirable** output buffering disabling:

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i;

  setvbuf(stdout, (char *)NULL, _IONBF, 0);
  setvbuf(stderr, (char *)NULL, _IONBF, 0);

  printf(‘‘Command: %s\n’’, argv[0]);

  for (i = 1; i < argc; i++)
  printf(‘‘Arg: %s\n’’, argv[i]);
```

```
  return 0;
}
```

5 Practice

1. Creating a console program in Eclipse:

   (a) Open the *File*, *New*, *C Project*.

   (b) Give the project a name and ensure that MinGW GCC is used for the toolchain.

   Project type should be *Executable*.

2. Creating new source files: *File*, *New*.

3. Usually, you want to create a source file, so choose *Source file*. When you name the file, it should have an extension of `.c`.

   Sometimes, you want a header file. At those times, choose *Header File*. The filename should have an extension of `.h`.

4. To run your program, open the *Run As* drop-down, select *Open Run Dialog*, then create a new *C/C++ Local Application*.

5. Practice program: Write a C program which accepts exactly three integer command-line arguments and prints the largest of them.

6. Another practice program using `malloc()`: Write a C program which takes a single integer as a command-line argument, creates an `int` array with that many elements, reads that many inputs from `stdin`, and the outputs the inputs in reverse order.