MINI LECTURE 2:
PROGRAM ARGUMENTS

COMP1002J: Introduction to Programming 2

Dr. Brett Becker (brett.becker@ucd.ie)

Beijing Dublin International College

Passing arguments to a function

- We are all familiar with passing arguments to a function
  - This allows us to get data ‘into’ a function from the ‘outside’
- But what about data that isn’t inside the program at all?
  - We just learned that we can read data from a file, but what if I don’t want to “hard-code” the name of the file in my program?
  - I could write a program to ask me the name of a file, and then read the data from that file.
  - But what if I want to *tell* my program what file, without being asked?

- This sounds like a job for a *program argument*

Passing Arguments To a Program

- C provides a way for you to pass arguments directly to a program from the command line. Here, we have a program called "display" that we want to pass some arguments to):

```
display argument1 argument2
```

- To make your program read this, you must modify the main() function:

```
int main(int argc, char **argv)
```

- The first parameter is the **number of arguments** and the second parameter is like an **array of strings**.

Passing Arguments To a Program

```
int main(int argc, char **argv)
```

- The first parameter is the **number of arguments** and the second parameter is like an **array of strings**.

Actually, this looks like a pointer to a pointer to me!

Passing Arguments To a Program

- The following program prints its arguments:

```
#include <stdio.h>

int main(int argc, char **argv)
{
  int i;
  for ( i=0; i<argc; i++ )
      printf( "Parameter %d is: %s\n", i, argv[i] );
}
```

- Notice how each argument is accessed by index, just like with a regular array.
- The difference is that this time, there is a string in each position of the array (actually a pointer to an array of characters)

file: arguments.c

Passing Arguments To a Program

```
printf( "Parameter %d is: %s\n", i, argv[i] );
```

- Notice how each argument is accessed by index, just like with a regular array.
- The difference is that this time, there is a string in each position of the array (actually a pointer to an array of characters)

So it's an array of character arrays
Or
An array of arrays
Or
A two-dimensional array!

Printing Arguments to a Program

- Run this code like this (for example):

```
arguments 1 2.0 B rett
```

- Output:

```
$>Parameter 0 is: arguments
  Parameter 1 is: 1
  Parameter 2 is: 2.0
  Parameter 3 is: B
  Parameter 4 is: rett
```

Arguments

- Notice that the first argument is the name of the program.
  - Actually, it is the command that was used to run the program.
- The other arguments are strings that were entered after the command (separated by spaces)
- Why is this useful?
  - Suppose I have a program that asks a user to enter a filename each time it runs.
  - If I run this program many times, I must type a filename every time, even if I want to use the same file.
  - With an argument, I can use the same command as last time much more easily.
  - Also, it is easier to run my program from within another program.

This program prompts a user for a filename, and then outputs the contents of that file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
   printf( "Enter a filename: "); // prompt user for filename
   char* str = malloc(80*sizeof(char)); //string for filename
   fgets (str, 80*sizeof(str), stdin); //get filename
   str[strlen(str)-1]='\0'; //replace trailing \n that fgets grabs
with \0
   FILE *fp = fopen( str, "r" );
   //check that file open succeeded
   int c = getc( fp );
   while( c != EOF ) {
      putchar( c );
      c = getc( fp );
   }
   fclose( fp );
}
```

files: fileread0.c
test.txt

Using Command-Line Arguments

- What if I didn’t want to ask the user for a file name, but instead specify it myself at runtime?

Using Command-Line Arguments

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
   if ( argc != 2 ) {
      printf( "No filename\n" );
      exit(1);
   }

   FILE *fp = fopen(argv[1], "r");
   if ( fp == NULL ) {
      printf("[%s] not opened\n",
             argv[1]);
      exit(1);
   }
```

```c
   int c = getc( fp );
   while( c != EOF ) {
      putchar( c );
      c = getc( fp );
   }

   fclose( fp );
}
```

file: fileread.c

Using Command-Line Arguments

```
$> fileread test.txt
```

Output

```
NVIDIA has revealed that Chinese internet giant Tencent will deploy Tesla GPUs in its public cloud
infrastructure. The GPUs are being deployed to help enterprise customers develop machine learning
services.
Based on revenue, Tencent is China's largest internet company, offering a variety of social media, e-
commerce, and gaming services for the world's fastest growing online market. The company took in
$15 billion in revenue in 2015, trailing only Amazon ($107B), Google ($75B), and Facebook ($18B).
Tencent’s market value that year was $206 billion.
The company's public cloud is aimed at online businesses and other enterprise customers that have
turned to the utility model for their computing infrastructure. The Tesla GPUs – including both the
P100 and P40 – will be made available to these customers via the Tencent cloud, who are building
machine learning services, either for their own customer base or for internal use. These include such
things as natural language processing, facial recognition, automated customer service, and supply
chain logistics.
"Tencent Cloud GPU offerings with NVIDIA's deep learning platform will help companies in China
rapidly integrate AI capabilities into their products and services," said Sam Xie, vice president of
Tencent Cloud, in prepared statement. "Our customers will gain greater computing flexibility and
power, giving them a powerful competitive advantage.
Tesla GPUs have become the de facto standard for machine learning workloads for hyperscale
companies. The P100, in particular, is NVIDIA's most powerful neural network training processor to
date, and is being deployed in both AI-oriented supercomputers and public cloud infrastructure. It's
general-purpose enough to handle both machine learning workload and more traditional HPC and
data analytics applications.
The P40 is more precisely targeted to the machine learning space, and is specifically aimed at the
inferencing side of those applications. In December 2016, Tencent had launched servers with M40
GPUs, the Maxwell-generation version of the P40.
Tencent, by the way, has also deployed IBM Power servers in its hyperscale datacenters for “big
data” workloads. Theoretically, Tencent could deploy the NVIDIA Tesla GPUs in such servers, which
would leverage the built-in NVLink support included in the latest Power processors.  This would
accelerate data communications between the host CPU (Power8) and the GPU (the P100), and offer
Tencent's enterprise customers a premier platform for machine learning.
More likely though, the GPUs are being hooked up to Intel Xeon processors in a more traditional
server setup. According to NVIDIA, the Tencent cloud servers will incorporate up to eight GPUs in
order to maximize performance for these machine learning workloads. Deployment is scheduled for
the first half of this year.
Source: https://www.top500.org/news/tencent-will-outfit-its-public-cloud-with-latest-nvidia-tesla-gpus/
```