CS265
Advanced Programming Techniques

C Command Line Arguments

Command Line Arguments in C Programs

- When main has **command line arguments** it is defined using

```
int main(int argc, char *argv[])
{
        …
}
```

where

`argc`	is the "argument count", the number of arguments
`argv`	is the "argument vector", an array of pointers to the arguments (stored as strings)
`argv[0]`	points to the name of the program
`argv[1]` through `argv[argc-1]`	point to the remaining arguments
`argv[argc]`	is always a **null pointer**

Alternate Definition

- We can write

```
int main(int argc, char *argv[])
{
        …
}
```

- Or we can write the alternate but equivalent Why ??

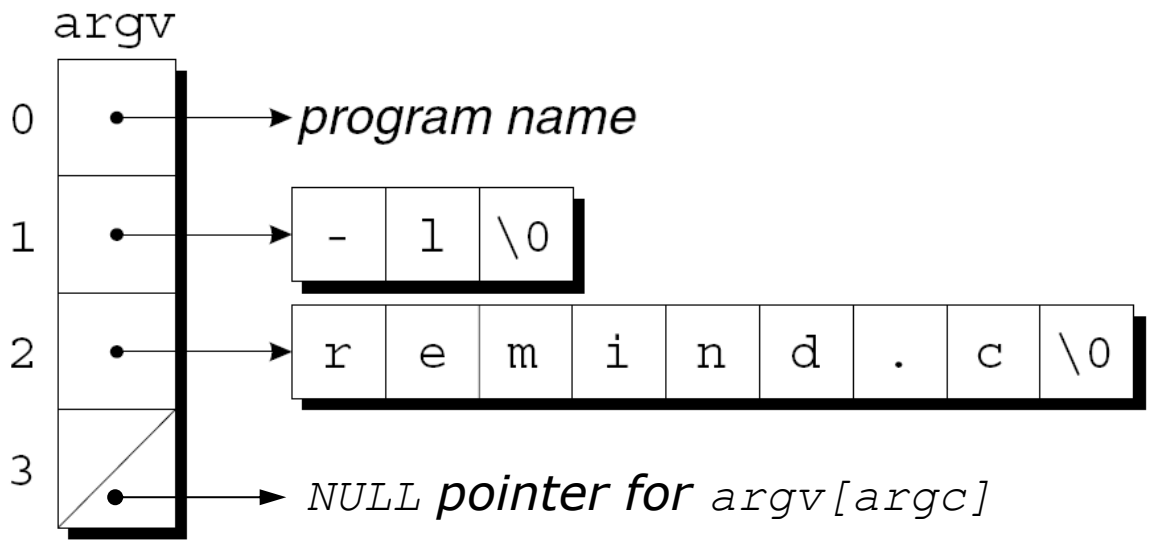
```
int main(int argc, char **argv)
{
        …
}
```

Command-Line Arguments

- If the user enters the command line

  `ls -l remind.c`

  then `argc` will be 3, and `argv` will have the following appearance:



Accessing Command Line Arguments using an array

To traverse the arguments, use an integer variable as an index into the `argv` array:

```
#include <stdio.h>

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

   printf("argc = %d \n", argc);
   printf("argv[0] = %s \n", argv[0]);

   for (int i=1; i<argc; i++)
      printf("argv[%d] = %s \n", i, argv[i]);

}
```

```
$ a.out
argc = 1
argv[0] = a.out
$ a.out 1 2 3
argc = 4
argv[0] = a.out
argv[1] = 1
argv[2] = 2
argv[3] = 3
```

Accessing Command Line Arguments using a double pointer

To traverse the arguments, set up a pointer to `argv[1]`, then increment the pointer repeatedly:

```
#include <stdio.h>

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

   printf("argc = %d \n", argc);
   printf("argv[0] = %s \n", argv[0]);

   char **p;
   for (p = &argv[1]; *p != NULL; p++)
      printf("%s\n", *p);

}
```

```
$ a.out
argc = 1
argv[0] = a.out
$ a.out 1 2 3
argc = 4
argv[0] = a.out
argv[1] = 1
argv[2] = 2
argv[3] = 3
```

Command Line Arguments

- Since `argv` is an array of pointers, accessing command-line arguments is easy.
- One way to do this is to use an integer variable as an index into the `argv` array:

```
int i;

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

- Another technique is to set up a pointer to `argv[1]`, then increment the pointer repeatedly:

```
char **p;

for (p = &argv[1]; *p != NULL; p++)
  printf("%s\n", *p);
```

Lessons

- Lesson 1: Learn C to become a power programmer
- Lesson 2: C / C++ are the defacto systems programming languages

Resources

- These notes
- *C Programming: A modern Approach* by K.N. King, 2008
- Chapter 13