Command Line Arguments

Section 3.4

Introduction

- In Unix, it's common to run programs from the command-line
  - *Command Prompt$* **nano program.c**
  - *Command Prompt$* **ls -la**
- These commands themselves are known as **command line arguments**
- We can access them from within our C programs

Command Line Arguments

- **Command line arguments** allow the user to specify how the they want to use the program
- Common for Unix commands to be run with command line arguments
    - **vi prog1.c**
    - **ls -la**
- Often specify an option for the program or a file to work on

Example

- Consider the following command
  - **ls -l -t**
    - **-l** lists the items vertically
    - **-t** sorts by time modified
  - Three command line arguments

main() Function

- For our program to accept command line arguments, the main function needs to have a couple parameters
  - int main(**int argc**, **char *argv[]**)

- Command line arguments are the inputs into the main function of the program
  - int main(**int argc, char *argv[]**)
- Java also supports command line arguments
  - public static void main(**String[] args**)

argc and argv

- **argc** and **argv** are a way of getting command line arguments into our programs
    - Command line arguments are switches and other things that come after a Unix command when it is run
    - The command itself is also included as a command line argument

argc

- **argc** stands for **argument count**
- Tells us how many command line arguments were given when our program was run
  - Name of the program is included in the count

- Command line arguments are text (strings)
    - Even if they're numbers, they're stored as characters
- **argc** is a number that tells us how many strings there are
    - In Java, the equivalent would be args.length
- **argc** is the size of **argv** (how many elements **argv** has)

- Examples:
  - **ls -l -a**
    - **argc** is 3
  - **nano program1.c**
    - **argc** is 2
  - **rm -r /**
    - **argc** is 3
  - **ls**
    - **argc** is 1

argv

- **argv** stands for **argument vector**
- **argv** is an array of strings that contains the command line arguments
    - Since a string is an array of chars in C, argv is a 2D array of chars

- Since basically an array of strings, each string is **argv[*i*]**, for some index ***i***
- Or could be thought of as a 2D array of characters
    - Character from **argv[*i*]** would be **argv[*i*][*j*]**, for some index ***j***

Accessing argv Strings

- Each element of **argv** is a string
  - Just need to index into **argv** to access command line arguments
  - **argv[0]** is the command itself
  - If there is a second command line argument, it will be **argv[1]**

- Examples:
    - **ls -l -a**
        - **argv[0]** is ls, **argv[1]** is -l, **argv[2]** is -a
    - **nano program1.c**
        - **argv[0]** is nano, **argv[1]** is program1.c
    - **rm -r /**
        - **argv[0]** is rm, **argv[1]** is -r, **argv[2]** is /
    - **ls**
        - **argv[0]** is ls, there is no **argv[1]** here

Accessing argv Characters

- Since a string is an array of characters in C, we can access the individual characters of an element of argv
- Second pair of [ ] allow us to access the characters
  - **argv[0]** is a string
  - **argv[0][0]** is a char (first char of **argv[0]**)

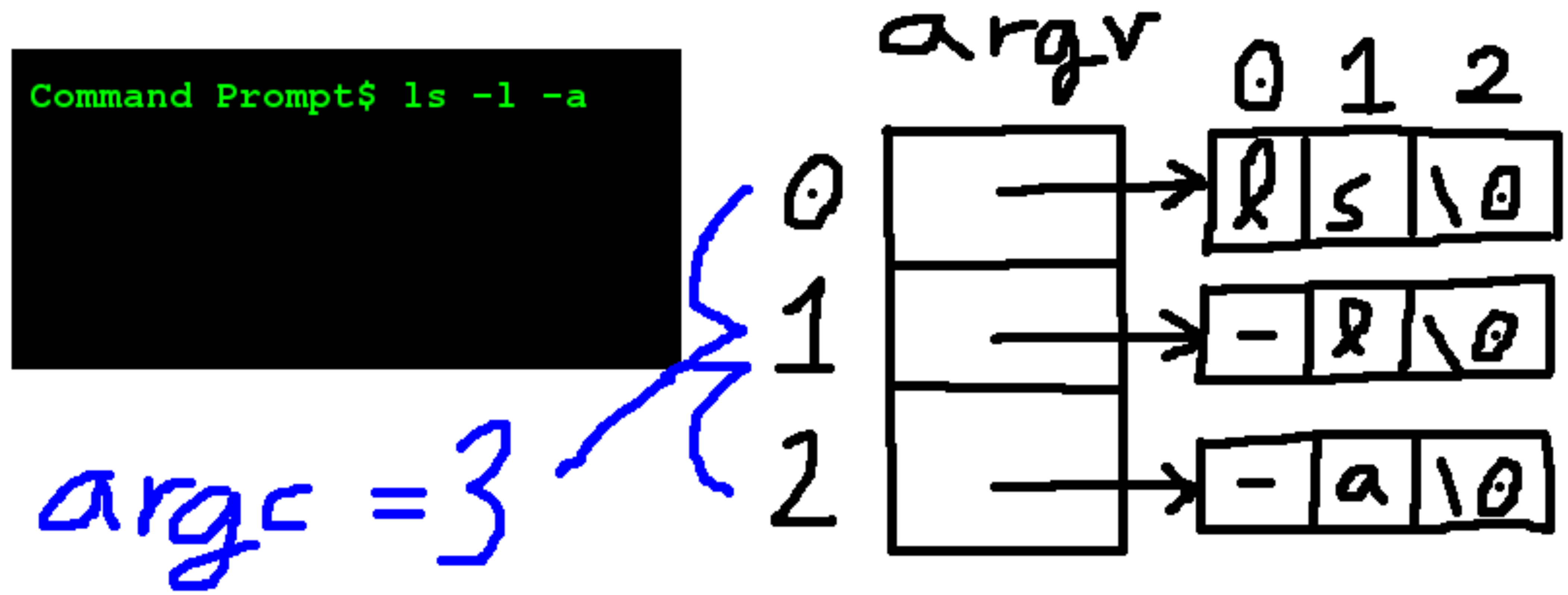
- Examples:
  - **ls -l -a**
    - **argv[0][0]** is l
    - **argv[0][1]** is s
    - **argv[1][0]** is -
    - **argv[1][1]** is l
    - **argv[2][0]** is -
    - **argv[2][1]** is a

Address Label(s)	Label	Address	Value
&(argc)	argc	400 – 403	3
&(argv[0][0]) or **argv[0]**	argv[0][0]	404	'l'
&(argv[0][1])	argv[0][1]	405	's'
&(argv[0][2])	argv[0][2]	406	'\0'
&(argv[1][0]) or **argv[1]**	argv[1][0]	407	'-'
&(argv[1][1])	argv[1][1]	408	'l'
&(argv[1][2])	argv[1][2]	409	'\0'
&(argv[2][0]) or **argv[2]**	argv[2][0]	410	'-'
&(argv[2][1])	argv[2][1]	411	't'
&(argv[2][2])	argv[2][2]	412	'\0'

Processing Command Line Arguments

- To process command line arguments, can use a for loop
  - Start counting at 0 and count up to argc, the number of arguments present
- Next slide shows how to print all the command line arguments

```
for (i = 0; i < argc; i++)
{
    printf("%s\n", argv[i]);
}
```



```
Command Prompt$ ls -l -a
```

argv[0] is ls

argr[0][0] is l