Hijjawi Faculty for Engineering Technology

Computer Engineering Department

System Programming Lab (Cpe 466)

Lab 1: Introduction

Goals:

- Introduce *C* and programming under Linux using gcc compiler.
- How to pass parameters to programs as command line arguments.

Background:

Almost all shell commands expect some parameters that are usually used to alter the behavior of that command. The number of parameters and their types is variable but is usually known for each command.

One way to solve this problem is to have multiple instances of the command compiled with a fixed number of parameters (e.g. `ls`; `lsl`, `lsla`,…). This method is not practical, inefficient and does not always work. A more practical method is to write a single command that expects parameters to be passed from the command line. The programmer, with help of the OS, should pars these parameters and performs the required functionality.

The lab introduces the concept of passing parameters to programs from the command line. These parameters are passed through arguments declared in the main function and they become part of its scope. The main function should be declared in a form shown in help page.

Libraries will be needed :

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

Functions will be used:

```
int main (int argc, char ** argv , char ** envp);
int  atoi(const char *); see the help sheet.
```

Required Task:

- Write a *C* program that reads a variable number of parameters, prints them and their count.

- Write a *C* program that implements a simple command line calculator. Its first parameter is a mathematical operation (+,-, x, /) that will be performed on the rest of the parameters.

- Write a *C* program that implements a simple command to calculate area for three - shapes. Its first parameter is the 1st letter of shape's name, the rest of parameters will be different for each shape .Square(s) needs one parameter ,rectangle (r) needs two parameters (length and width) ,circle(c) need one parameter (radius).

Good Luck

Help for you

Main

int main (int argc, char ** argv , char ** envp); or

int main (int argc, char * argv [ ], char * envp[ ]);
Where :
argc: number of argments passed including the command itself
argv: NULL terminated array of strings represents one parameter staring with a command
envp: NULL terminated array of strings represents one environment variable.

The parameters `argc`, *argument count*, and `argv`, *argument vector*,[1] respectively give the number and value of the program'scommand-line arguments. The names of `argc` and `argv` may be any valid identifier in C, but it is common convention to use these names.

---

atoi()
convert a string to an integer

SYNOPSIS

```
#include <stdlib.h>
int atoi(const char *nptr);
```

DESCRIPTION

Converts a string representing an integer to an integer. The string to be converted consists of optional leading white space, an optional sign and a series of contiguous digits.

The call `atoi(`*nptr*`)` is equivalent to:

```
(int) strtol(nptr,(char **)NULL, 10)
```

PARAMETERS

nptr
Points to a string to be converted to an integer.

RETURN VALUES

The `atoi()` function returns the converted integer value.

If the converted value overflows, `LONG_MIN` or `LONG_MAX` is returned (according to the sign of the value) and `errno` is set to `ERANGE`. If no character could be converted, zero is returned.