Queen Mary University of London

School of Electronic Engineering and Computer Science

ECS501U – C Programming
Laboratory Session Week_9

Learning Objectives

- To produce an example of a UDP server that uses *struct hostent* structure to identify clients
- To produce examples of exec() usage
- To produce examples of fork() usage

Exercises

You should attempt the exercises below by using only the C constructs that you learnt up to teaching week 9, and:

1. Write pseudo code to describe the required algorithm to solve the exercise (or draw up a flowchart), before writing and testing the actual code.
2. Add comments to your code.
3. Make your code neat, by using indentation and parenthesis (where appropriate).
4. Give meaningful names to functions and variables.

Exercise 1

Write a UDP server similar to the example given in class, but the server will perform the conversion of a binary number to decimal, this is the **converter binary to decimal server**. The UDP client will simply send a binary number and in response will receive the equivalent decimal number calculated by the server. The server before doing the conversion for the client, identifies the client by name and IP address using the `struct hostent` and the function `gethostbyaddr()`. **Hint:** Use the UDP server and client code of the lecture example as the first skeleton of this exercise and then make the necessary changes incrementally. The code for the conversion from binary to decimal is given in Figure 1. You also will need to use the function `atol()` to convert a string to a long int (`long int atol(const char *str)`) and the function `sprintf()` to convert an integer value (decimal number) to a string (`int sprintf (char *string, const char *form, … ); `)[1].

[1] Example of usage of `sprinf():sprintf(str, "Value of Pi = %f", pi_value);`

```
…
#include <math.h>

power(int i)
{
 int j,p=1;
 for(j=1;j<=i;j++)
 p = p*2;
 return(p);
}

int covertBinToDec (long int n)
{
  int x,s=0,i=0,flag=1;
  while(flag==1)
   {
     x=n%10;
     s=s+x*power(i);
     i=i+1;
     n=n/10;
     if(n==0)
        flag=0;
   }
  return s;
}
```

Figure 1

Exercise 2

The program in Figure 2 displays the network configuration of the current machine by using the `exec()` family of functions. The command to display the network configuration on Linux and Mac machines is **`"/sbin/ifconfig"`**. On Windows, the equivalent command is ***`"ipconfig"`*** that is stored somewhere on the command path. Complete the missing parts of the program shown in Figure 2 and test your program. **Hint:** Choose a suitable `exec()` function you have seen in class and have a look at Chapter 9 of Head First C.

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

int main()
{
   /* try to run the "/sbin/ifconfig" program firstly
   if (.............................){
   /* if failed, try the “ipconfig” command
      if (...........................){
          fprintf(stderr,"Cannot run ipconfig: %s",strerror(errno));
          return 1;
      }
   }
   return 0;
}
```

Figure 2

Exercise 3

Write a program to execute the ***changeUser*** program given in Figure 3 to change the value of the environment variable **USER** to **"ECS501U"** (instead of the value of this variable set in your computer). Your program needs to display the value of **USER** before and after the change. **Hint:** First look at the similar example given in class, then make your program first display the current value of **USER** that is the value of the environment variable in your computer, then you need to create a new environment variable for **USER** using, for example, `char *my_env[],` and then choose one function from the `exec()` family to display the new value.

```
/* This is the code of changeUser.c */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
   printf("\n… After exec: USER=%s\n",getenv("USER"));

   return 0;
}
```

Figure 3

Exercise 4

Write a program that performs a `fork()` and then invokes the ***runChild*** function (given in Figure 4) in the child process, and five seconds later, the parent prints a message saying, **"I'm still here!"** and the **process id** of both parent and child processes. Use the ***sleep*** library function to put the process to sleep for five seconds.

```
void runChild(void)
{
   printf("\n I am the child! My pid is %d\n",getpid());

}
```

Figure 4

ECS501U – END of LAB Week_9