LAB ASSIGNMENT-1

CSN-361

Submitted by:
Ritik Kumar 17114063

Problem Statement 1:

Write a C program in the UNIX system that creates two children and four grandchildren (two for each child). The program should then print the process-IDs of the two children, four grandchildren and the parent in this order.

Algorithm:

Run the program to get a process with a PID.
Fork that process to get 2 children. Print the PIDs of both of them.
For each of them, fork 2 grand-child and print their PIDs
Wait till both children are done and print the parent.

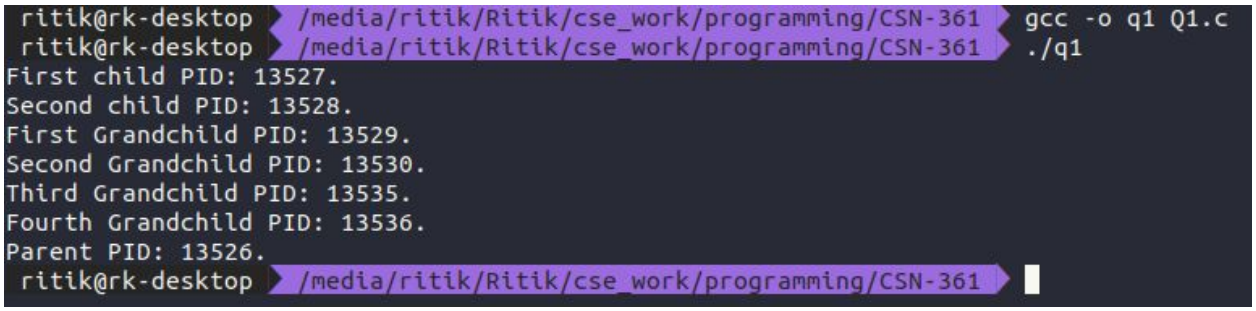
Data Structures:

Integers to store PIDs of all 7 processes.

Code

```c
/** @file Q1.c
 *  @brief Q1 get childs and grandchilds of a process.
 *
 *  @author Ritik
 */

#include <stdio.h>
#include <unistd.h>
#include<sys/wait.h>

/** @brief Q1 entrypoint.
 */
int main()

{
  int pid = -1, pid1 = -1, pid2 = -1, pid11 = -1, pid12 = -1, pid21 = -1,
    pid22 = -1;
  int ppppid = getpid();

  pid1 = fork(); // child1
  pid2 = fork(); //child2

  if(pid1>0 && pid2>0){ //parent
    printf("First child PID: %d.\n", pid1);
    printf("Second child PID: %d.\n", pid2);
  }
  else if(pid1==0 && pid2>0)
  { // child1
    pid11 = pid2;
    pid12 = fork();

    if(pid12 != 0){ // child1
      printf("First Grandchild PID: %d.\n", pid11);
      printf("Second Grandchild PID: %d.\n", pid12);
    }
  }

  else if(pid2==0 && pid1!=0){
    int i = 1000000;
    while (i > 0){
      i--;
    }
    pid21 = fork();
    if(pid21 != 0){ //child2

      pid22 = fork();
      if(pid22 != 0){ //child2

        printf("Third Grandchild PID: %d.\n", pid21);
        printf("Fourth Grandchild PID: %d.\n", pid22);

        printf("Parent PID: %d.\n", ppppid);
      }
    }
  }
}
```

Running Code



Problem Statement 2:

Write a C++ program to print the MAC address of your computer.

Algorithm:

Create a Struct to store Network devices
Create a Socket and store the FD
Store the network device name in the struct
Fetch and store the MAC address using the *ioctl* system call
Close the Socket
Print all the segments of *ifreq* separately using ':'

Data Structures:

Int fd: File descriptor
struct ifreq ifr: Store the network device info

Code

```cpp
/** @file Q2.cpp
 *  @brief Q2 Get the MAC address of the interface using
 *         <command> <devicename>
 *
 *  @author Ritik
 */

#include <iostream>
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <net/if.h>
#include <unistd.h>

using namespace std;

/** @brief Q2 entrypoint.
 *
 *  @param argc Count of the arguments
 *  @param argv Array of parameters
 */
main(int argc,char **argv){

    if(argc != 2){
        fprintf(stderr,"usage: <command> <devicename>\n");
        exit(1);
    }

    unsigned char uc_Mac[32]={0};

    int fd;
    struct ifreq ifr;
    char *iface = argv[1];
    char *mac;

    fd = socket(AF_INET, SOCK_DGRAM, 0);

    ifr.ifr_addr.sa_family = AF_INET;
    strncpy((char *)ifr.ifr_name , (const char *)iface , IFNAMSIZ-1);

    ioctl(fd, SIOCGIFHWADDR, &ifr);

    close(fd);

    printf("MAC Address for the device %s : %02x:%02x:%02x:%02x:%02x:%02x\n",
        argv[1],
      (unsigned char) ifr.ifr_hwaddr.sa_data[0],
      (unsigned char) ifr.ifr_hwaddr.sa_data[1],
      (unsigned char) ifr.ifr_hwaddr.sa_data[2],
      (unsigned char) ifr.ifr_hwaddr.sa_data[3],
      (unsigned char) ifr.ifr_hwaddr.sa_data[4],
      (unsigned char) ifr.ifr_hwaddr.sa_data[5]);

    return 0;
```

Running Code

Problem Statement 3:

Write your own version of the ping program in C language.

Algorithm:

Input syntax: *sudo ./q3 <hostname> <times>*
Get the domain name and the number of pings
Convert the domain name to IP address
Create a socket file descriptor
Make a packet to be sent in ping with relevant information
Send a socket message wit the packet to the destination IP address at port 0
Read the received response from the server and print the length
Repeat n number of times asked
At every step, in case of an error, exit the program we suitable message

Data Structures:

Int: Socket File Descriper
int: Store number of times
char *: IP address of the destination
Struct icmp_hdr: Store Ping packet
struct sockaddr_in: Store destination information
Int response: store the response byte array

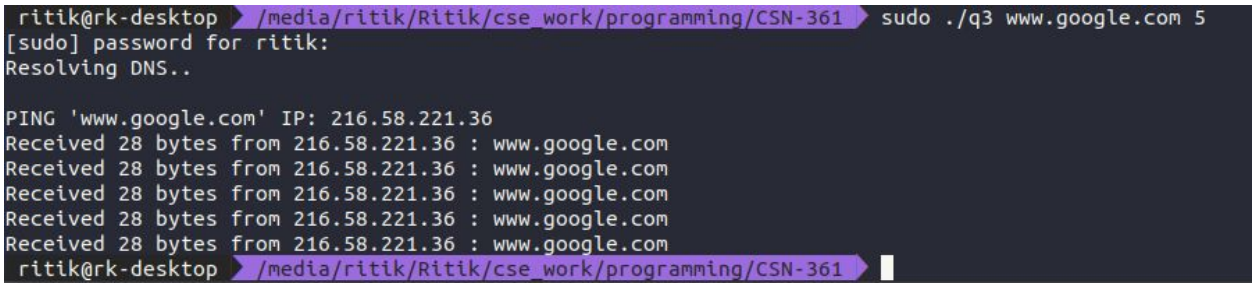
Code

```c
/** @file Q3.c
 *  @brief Q3 Ping a server. Use sudo <command> <hostname> <times>
 *
 *  @author Ritik
 */
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <unistd.h>
#include <netdb.h>
#include <string.h>

/** @brief Get the IP address from the domain name
 *
 * @param addr_host The domain name.
 * @return a char* to the ip address.
 */
char *dns_lookup(char *addr_host)
{
    printf("Resolving DNS..\n");
    struct hostent *host_entity;
    char *ip=(char*)malloc(NI_MAXHOST*sizeof(char));
    int i;

    if ((host_entity = gethostbyname(addr_host)) == NULL)
    {
        // No ip found for hostname
        return NULL;
    }

    strcpy(ip, inet_ntoa(*(struct in_addr *)
                         host_entity->h_addr));

    return ip;
}

/** @brief Q3 entrypoint.
 *
 *  @param argc Count of the arguments
 *  @param argv Array of parameters
 */
int main(int argc, char *argv[]) {

    int count = 1;
    char *ip_addr;

    if (getuid() != 0)
    {
        fprintf(stderr, "%s: root privelidges needed\n", *(argv + 0));
        exit(EXIT_FAILURE);
    }
```

```c
if(argc < 2)
{
    printf("\nIncorrect Format %s <address>\n", argv[0]);
    return 0;
}

if (argc == 3)
{
    if(atoi(argv[2]) != 0)
        count = atoi(argv[2]);
}

ip_addr = dns_lookup(argv[1]);

if(ip_addr==NULL)
{
    printf("\nCould not resolve hostname!\n");
    return 0;
}

printf("\nPING '%s' IP: %s\n", argv[1], ip_addr);

// Creating Socket
int s = socket(PF_INET, SOCK_RAW, 1);

if(s <= 0)
{
    perror("Socket Error");
    exit(0);
}

// Create the ICMP Struct Header
typedef struct {
    uint8_t type;
    uint8_t code;
    uint16_t chksum;
    uint32_t data;
} icmp_hdr;

icmp_hdr pckt;

// Set the appropriate values to our struct, which is our ICMP header
pckt.type = 8;          // The echo request is 8
pckt.code = 0;          // No need
pckt.chksum = 0xfff7;   // Fixed checksum since the data is not changing
pckt.data = 0;          // We don't send anything.

// Creating a IP Header from a struct that exists in another library

struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = 0;
addr.sin_addr.s_addr = inet_addr(ip_addr);
```

```c
    // Send our PING
    while(count > 0)
    {
        count--;
        int actionSendResult = sendto(s, &pckt, sizeof(pckt),
                                      0, (struct sockaddr*)&addr, sizeof(addr))
                                          ;

        if(actionSendResult < 0)
        {
            perror("Ping Error");
            exit(0);
        }

        // Prepare all the necessary variable to handle the response
        unsigned int resAddressSize;
        unsigned char res[30] = "";
        struct sockaddr resAddress;

        // Read the response from the remote host
        int response = recvfrom(s, res, sizeof(res), 0, &resAddress,
                                &resAddressSize);

        if( response > 0)
        {
            printf("Received %d bytes from %s : %s\n", response, ip_addr, argv[
                1]);
        }
        else
        {
            perror("Response Error");
            exit(0);
        }
    }
    return 0;
}
```

Running Code



Problem Statement 4:

Write a C program to find the hostname and the IP address of your computer.

Algorithm:

Create a Struct to store Network devices
Get the hostname using *gethostname* system call
Get the host information using *gethostbyname* system call
Create a socket and store its address in the struct
Store the network device name in the struct
Fetch and store the IP address using the *ioctl* system call
Close the Socket
Use *inet_aton* to convert the Internet host address cp from the IPv4 numbers-and-dots notation into binary form

Data Structures:

Int n: File descriptor
struct ifreq ifr: Store the network device info

Code

```c
/** @file Q4.c
 *  @brief Q4 get the host name and the IP address of your computer.
 *         <command> <devicename>
 *  @author Ritik
 */

#include <stdlib.h>
#include <errno.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <net/if.h>
#include <unistd.h>
#include <arpa/inet.h>

/** @brief Q4 entrypoint.
 *
 *  @param argc Count of the arguments
 *  @param argv Array of parameters
 */
int main(int argc, char *argv[]) {

   if(argc < 2)
   {
      fprintf(stderr,"usage: <command> <devicename>\n");
      return 0;
   }
   int n;
   struct ifreq ifr;
   char* array = argv[1];
   char host[256];
   struct hostent *host_entry;
   int hostname;

   // Get the host name
   hostname = gethostname(host, sizeof(host));
   if (hostname == -1) {
      perror("gethostname");
      exit(1);
   }

   // Get host information
   host_entry = gethostbyname(host);
   if (host_entry == NULL) {
      perror("gethostbyname");
      exit(1);
   }

   n = socket(AF_INET, SOCK_DGRAM, 0);

   //Type of address to retrieve - IPv4 IP address
   ifr.ifr_addr.sa_family = AF_INET;

   //Copy the interface name in the ifreq structure
   strncpy(ifr.ifr_name , array , IFNAMSIZ - 1);

   ioctl(n, SIOCGIFADDR, &ifr);
   close(n);

   //display result
```

Running Code