CSN - 361

Lab Assignment - 1

Bhavye Jain

17114020

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, the four grandchildren and the parent in this order.

SOLUTION:

The program uses the fork() system call to create child processes. The exit() call is used to terminate unwanted processes.

```
#include <bits/stdc++.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

using namespace std;

int main(){
  cout << "Parent process id: " << getpid() << endl << endl;
  for(int i = 0; i < 2; i++){
    if(fork() == 0){
      cout << "Child " << (i+1) << " with pid "<< getpid() << " from parent pid " << getppid() << 
endl;
      for(int j = 0; j < 2; j++){
        if(fork() == 0){
          cout << "Grandchild " << (i*2 + 1 + j) << " with pid "<< getpid() << " from parent pid " << 
getppid() << endl;
          exit(0);    // grandchild process terminates execution
        }
        wait(NULL);       // child waits for grandchildren to complete execution
      }

      exit(0);            // child process terminates execution
    }
    wait(NULL);    // parent process waits for child process to terminate execution
    cout << endl;
  }
  exit(0);
}
```

```
bhavye@ultron: ~/CSN361
File  Edit  View  Search  Terminal  Help
bhavye@ultron:~/CSN361$ g++ -o q1 q1.cpp
bhavye@ultron:~/CSN361$ ./q1
Parent process id: 15855

Child 1 with pid 15856 from parent pid 15855
Grandchild 1 with pid 15857 from parent pid 15856
Grandchild 2 with pid 15858 from parent pid 15856

Child 2 with pid 15859 from parent pid 15855
Grandchild 3 with pid 15860 from parent pid 15859
Grandchild 4 with pid 15861 from parent pid 15859

bhavye@ultron:~/CSN361$
```

Problem Statement 2

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

SOLUTION:

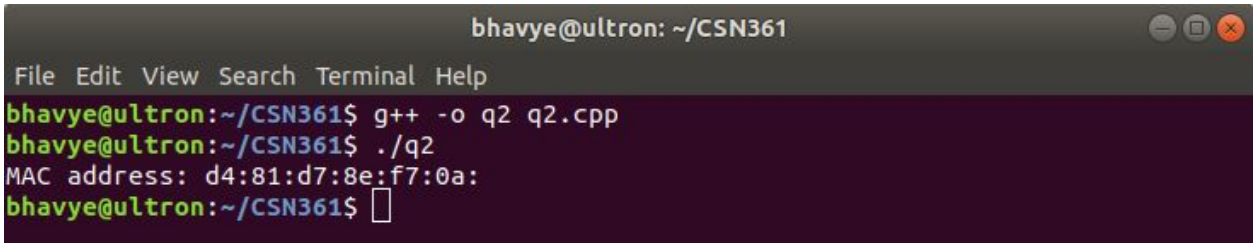
```
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/if.h>
#include <netdb.h>
#include <iostream>
#include <string.h>
#include <bits/stdc++.h>

using namespace std;

int main()
{
  struct ifreq ifr;
  int fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_IP);
  string ans;
  char buff[3];
  strcpy(ifr.ifr_name, "enp3s0");

  if (ioctl(fd, SIOCGIFHWADDR, &ifr) == 0) {
    for (int i = 0; i <= 5; i++){
      snprintf(buff, sizeof(buff), "%.2x", (unsigned char)ifr.ifr_addr.sa_data[i]);
      ans = ans + buff + ":";
    }
    cout << "MAC address: " << ans << endl;
    return 0;
  }

  return 1;
}
```


bhavye@ultron: ~/CSN361
File Edit View Search Terminal Help
bhavye@ultron:~/CSN361$ g++ -o q2 q2.cpp
bhavye@ultron:~/CSN361$ ./q2
MAC address: d4:81:d7:8e:f7:0a:
bhavye@ultron:~/CSN361$

Problem Statement - 3

Write your own version of ping program in C language.

SOLUTION:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <netinet/ip_icmp.h>
#include <time.h>
#include <signal.h>
#include <time.h>

#define PING_PKT_S 64
#define PORT_NO 0
#define PING_SLEEP_RATE 1000000
#define RECV_TIMEOUT 1

int pingloop=1;

struct ping_pkt {
  struct icmphdr hdr;
  char msg[PING_PKT_S-sizeof(struct icmphdr)];
};

unsigned short check_sum(void *b, int len) {
  unsigned short *buf = b;
  unsigned int sum = 0;
  unsigned short result;

  for (sum = 0; len > 1; len -= 2) sum += *buf++;
  if (len == 1)     sum += *(unsigned char*)buf;
  sum = (sum >> 16) + (sum & 0xFFFF);
  sum += (sum >> 16);
  result = ~sum;
  return result;
}
```

```c
void int_handler(int dummy) {
  pingloop=0;
}

char *dns_lookup(char *addr_host, struct sockaddr_in *addr_con) {
  printf("\nResolving DNS.....\n");
  struct hostent *host_entity;
  char *ip=(char*)malloc(NI_MAXHOST*sizeof(char));
  int i;

  if ((host_entity = gethostbyname(addr_host)) == NULL) return NULL;

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

  (*addr_con).sin_family = host_entity->h_addrtype;
  (*addr_con).sin_port = htons (PORT_NO);
  (*addr_con).sin_addr.s_addr = *(long*)host_entity->h_addr;

  return ip;
}

void ping_site(int ping_sock_fd, struct sockaddr_in *ping_addr,
char *ping_ip, char *rev_host) {
  int ttl_val = 64, msg_count = 0, i, addr_len,
  flag = 1, msg_received_count = 0;

  struct ping_pkt pckt;
  struct sockaddr_in r_addr;
  struct timespec time_start, time_end, tfs, tfe;
  long double rtt_msec=0, total_msec=0;
  struct timeval tv_out;
  tv_out.tv_sec = RECV_TIMEOUT;
  tv_out.tv_usec = 0;

  clock_gettime(CLOCK_MONOTONIC, &tfs);

  if (setsockopt(ping_sock_fd, SOL_IP, IP_TTL,
  &ttl_val, sizeof(ttl_val)) != 0) {
    printf("\nSetting socket options to TTL failed!\n");
    return;
  } else printf("\nSocket set to TTL..\n");

  setsockopt(ping_sock_fd, SOL_SOCKET, SO_RCVTIMEO,
  (const char*)&tv_out, sizeof tv_out);
```

```
while(pingloop) {
  flag = 1;

  bzero(&pckt, sizeof(pckt));

  pckt.hdr.type = ICMP_ECHO;
  pckt.hdr.un.echo.id = getpid();

  for (i = 0; i < sizeof(pckt.msg) - 1; i++) pckt.msg[i] = i+'0';

  pckt.msg[i] = 0;
  pckt.hdr.un.echo.sequence = msg_count++;
  pckt.hdr.checksum = check_sum(&pckt, sizeof(pckt));

  usleep(PING_SLEEP_RATE);

  clock_gettime(CLOCK_MONOTONIC, &time_start);
  if (sendto(ping_sock_fd, &pckt, sizeof(pckt), 0,
  (struct sockaddr*) ping_addr,
  sizeof(*ping_addr)) <= 0) {
  printf("\nPacket Sending Failed!\n");
  flag=0;
}

addr_len=sizeof(r_addr);

if (recvfrom(ping_sock_fd, &pckt, sizeof(pckt), 0,
(struct sockaddr*)&r_addr, &addr_len) <= 0
&& msg_count>1) {
  printf("\nPacket receive failed!\n");
} else {
  clock_gettime(CLOCK_MONOTONIC, &time_end);

  double timeElapsed = ((double)(time_end.tv_nsec - time_start.tv_nsec))/1000000.0;
  rtt_msec = (time_end.tv_sec - time_start.tv_sec) * 1000.0 + timeElapsed;

  if (flag) {
    if (!(pckt.hdr.type ==69 && pckt.hdr.code==0)) {
      printf("Error..Packet received with ICMP type %d code %d\n",
      pckt.hdr.type, pckt.hdr.code);
    } else {
      printf("%d bytes from %s (%s) rtt = %Lf ms.\n",
      PING_PKT_S, rev_host,
      ping_ip, rtt_msec);
      msg_received_count++;
```

```
      }
      }
    }
  }
  clock_gettime(CLOCK_MONOTONIC, &tfe);
  double timeElapsed = ((double)(tfe.tv_nsec - tfs.tv_nsec))/1000000.0;

  total_msec = (tfe.tv_sec-tfs.tv_sec)*1000.0 + timeElapsed;

  printf("\n===%s ping statistics===\n", ping_ip);
  printf("\n%d packets sent, %d packets received, %f percent packet loss. Total time: %Lf ms.\n\n",
  msg_count, msg_received_count,
  ((msg_count - msg_received_count)/msg_count) * 100.0, total_msec);
}

int main(int argc, char *argv[]) {
int sock_fd;
char *ip_addr, *reverse_hostname;
struct sockaddr_in addr_con;
int addrlen = sizeof(addr_con);
char net_buf[NI_MAXHOST];

if (argc != 2) {
printf("\nFormat %s <address>\n", argv[0]);
return 0;
}

ip_addr = dns_lookup(argv[1], &addr_con);
if (ip_addr == NULL) {
printf("\nDNS lookup failed! Could not resolve hostname!\n");
return 0;
}

sock_fd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
if (sock_fd < 0) {
printf("\nSocket file descriptor not received!!\n");
return 0;
} else printf("\nSocket file descriptor %d received\n", sock_fd);

signal(SIGINT, int_handler);

ping_site(sock_fd, &addr_con, ip_addr, argv[1]);
return 0;
}
```

```
bhavye@ultron: ~/CSN361
File  Edit  View  Search  Terminal  Help
bhavye@ultron:~/CSN361$ gcc -o q3 q3.c
bhavye@ultron:~/CSN361$ sudo ./q3 google.com

Resolving DNS.....

Socket file descriptor 3 received

Socket set to TTL..
64 bytes from google.com (172.217.167.46) rtt = 41.925271 ms.
64 bytes from google.com (172.217.167.46) rtt = 41.843706 ms.
64 bytes from google.com (172.217.167.46) rtt = 41.872164 ms.
64 bytes from google.com (172.217.167.46) rtt = 41.837022 ms.
64 bytes from google.com (172.217.167.46) rtt = 41.885591 ms.
^C64 bytes from google.com (172.217.167.46) rtt = 42.118856 ms.

===172.217.167.46 ping statistics===

6 packets sent, 6 packets received, 0.000000 percent packet loss. Total time: 56
69.929054 ms.

bhavye@ultron:~/CSN361$
```

Problem Statement - 4

Write a C program to find the host name from IP address.

SOLUTION:

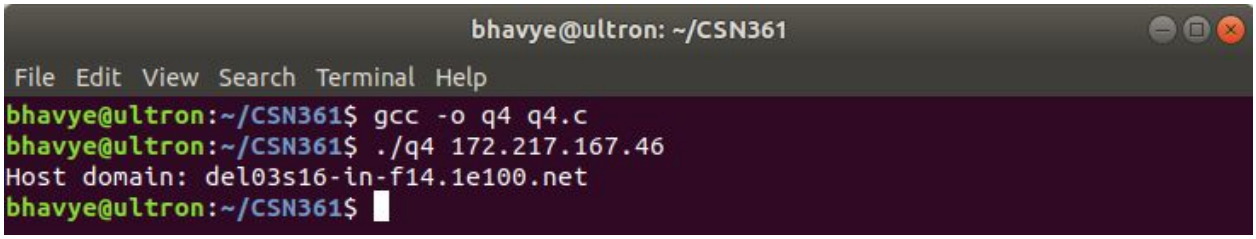
```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

char *host_lookup(char *ip_addr) {

  struct sockaddr_in temp;
  socklen_t l;
  char buf[NI_MAXHOST], *ret_buf;

  temp.sin_family = AF_INET;
  temp.sin_addr.s_addr = inet_addr(ip_addr);
  l = sizeof(struct sockaddr_in);
  if (getnameinfo((struct sockaddr *) &temp, l, buf, sizeof(buf), NULL, 0, NI_NAMEREQD)) {
    printf("Could not resolve lookup of the hostname\n");
    return NULL;
  }
  ret_buf = (char *)malloc((strlen(buf) + 1) * sizeof(char));
  strcpy(ret_buf, buf);
  return ret_buf;
}

int main(int argc, char *argv[]) {
  char *ip_addr = argv[1];
  char *reverse_hostname = host_lookup(ip_addr);
  printf("Host domain: %s\n", reverse_hostname);
}
```


bhavye@ultron: ~/CSN361
File Edit View Search Terminal Help
bhavye@ultron:~/CSN361$ gcc -o q4 q4.c
bhavye@ultron:~/CSN361$ ./q4 172.217.167.46
Host domain: del03s16-in-f14.1e100.net
bhavye@ultron:~/CSN361$