Computer Networks Laboratory CSN361

Lab Assignment 1

Ashutosh Chaubey (17114015)

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 function to get a new process with a Process ID.
- Fork that process to get 2 children. Print the Process IDs of both of them.
- For each of them, fork 2 grand-child and print their Process IDs
- Wait till both children are done and print the parent.

Data Structures:

Integers to store PIDs of all 7 processes.

Code:

surreal24_to_ours17_viewnorm.py | transform_util.py — commons | problem1.c | problem2.cpp | probler

```c
/** @file problem1.c
 *  @brief Problem Statement 1 : Prints process_ids of created child and grandchild processes
 *
 *  @author Ashutosh Chaubey
 */

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

/** @brief Problem Statement 1 entrypoint.
 */
int main()

{
  int process_id = -1, process_id1 = -1, process_id2 = -1, process_id11 = -1, process_id12 = -1, process_id21 = -1, process_id22 = -1;
  int new_process_id = getpid();

  process_id1 = fork(); // child1
  process_id2 = fork(); // child2

  if(process_id1>0 && process_id2>0){ //parent
    printf("1st child process_id: %d..(%d).\n2nd child process_id: %d.\n", process_id1, new_process_id, process_id2);
  }
  else if(process_id1==0 && process_id2>0)
  { // child1
    process_id11 = process_id2;
    process_id12 = fork();

    if(process_id12 != 0){ // child1
      printf("1st Grandchild process_id: %d.\n2nd Grandchild process_id: %d.\n", process_id11, process_id12);
    }
  }

  else if(process_id2==0 && process_id1!=0){
    int i = 1000000;
    while (i > 0){
      i--;
    }
    process_id21 = fork();
    if(process_id21 != 0){ //child2

      process_id22 = fork();
      if(process_id22 != 0){ //child2

        printf("3rd Grandchild process_id: %d.\n4th Grandchild process_id: %d.\n", process_id21, process_id22);

        printf("Parent process_id: %d.\n", new_process_id);
      }
    }
  }
}
```

Output

```
(base) djikstra@helios:~/Academic/CSN361/L1$ gcc problem1.c -o problem1
(base) djikstra@helios:~/Academic/CSN361/L1$ ./problem1
1st child process_id: 6682..(6681).
2nd child process_id: 6683.
1st Grandchild process_id: 6684.
2nd Grandchild process_id: 6685.
(base) djikstra@helios:~/Academic/CSN361/L1$ 3rd Grandchild process_id: 6686.
4th Grandchild process_id: 6687.
Parent process_id: 6681.
^C
(base) djikstra@helios:~/Academic/CSN361/L1$
```

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:

surreal24_to_ours17_viewnorm.py | transform_util.py — commons | problem1.c | problem2.cpp

```cpp
/** @file problem2.cpp
 *  @brief Problem Statement 2 Print mac address of computer , usage: <command> <devicename>
 *
 *  @author Ashutosh
 */

#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 Problem Statement 2 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 ur_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 your 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;
}
```

Output

```
(base) djikstra@helios:~/Academic/CSN361/L1$ g++ problem2.cpp -o problem2
(base) djikstra@helios:~/Academic/CSN361/L1$ ./problem2 dev1
MAC Address for your device dev1 : 61:[redacted]:fc:7f:00:00
(base) djikstra@helios:~/Academic/CSN361/L1$
```

Problem Statement 3

Write your own version of 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:

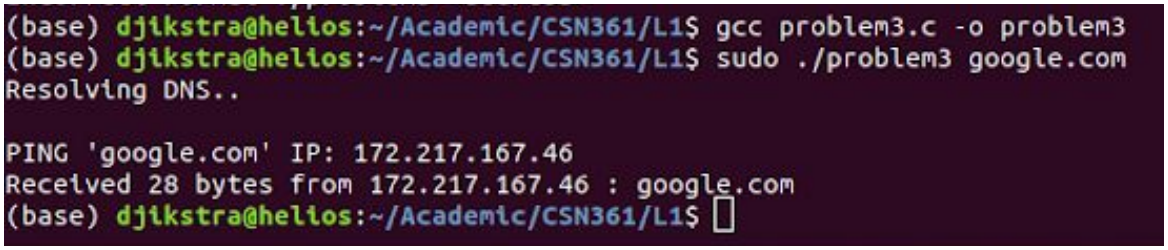
surreal24_to_ours17_viewnorm.py | transform_util.py — commons | problem1.c | problem2.cpp | problem3.c

```c
/** @file problem3.c
 *  @brief Problem Statement 3 Ping a server. Use sudo <command> <host_name> <times>
 *
 *  @author Ashutosh
 */
#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 Function to get the IP address from the domain name
 *
 * @param host_addr The domain name.
 * @return a char* to the ip address.
 */
char *domain_name_server_lookup(char *host_addr)
{
    printf("Resolving DNS..\n");
    struct hostent *host_entity;
    char *ip=(char*)malloc(NI_MAXHOST*sizeof(char));
    int i;

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

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

    return ip;
}

/** @brief Problem Statement 3 entrypoint.
 *
 *  @param argc Count of the arguments
 *  @param argv Array of parameters
 */
int main(int argc, char *argv[]) {

    int count = 1;
    char *ip_address;

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

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

```

```
60
61      if (argc == 3)
62      {
63          if(atoi(argv[2]) != 0)
64              count = atoi(argv[2]);
65      }
66
67      ip_address = domain_name_server_lookup(argv[1]);
68
69      if(ip_address==NULL)
70      {
71          printf("\nCould not resolve host_name!\n");
72          return 0;
73      }
74
75      printf("\nPING '%s' IP: %s\n", argv[1], ip_address);
76
77      // Creating Socket
78      int s = socket(PF_INET, SOCK_RAW, 1);
79
80      if(s <= 0)
81      {
82          perror("Socket Error");
83          exit(0);
84      }
85
86      // Create the ICMP Struct Header
87      typedef struct {
88          uint8_t type;
89          uint8_t code;
90          uint16_t chksum;
91          uint32_t data;
92      } icmp_hdr;
93
94      icmp_hdr pckt;
95
96      // Set the appropriate values to our struct, which is our ICMP header
97      pckt.type = 8;          // The echo request is 8
98      pckt.code = 0;          // No need
99      pckt.chksum = 0xfff7;   // Fixed checksum since the data is not changing
100     pckt.data = 0;          // We don't send anything.
101
102     // Creating a IP Header from a struct that exists in another library
103
104     struct sockaddr_in addr;
105     addr.sin_family = AF_INET;
106     addr.sin_port = 0;
107     addr.sin_addr.s_addr = inet_addr(ip_address);
108
109     // Send our PING
110     while(count > 0)
111     {
112         count--;
113         int actionSendResult = sendto(s, &pckt, sizeof(pckt),
114                                       0, (struct sockaddr*)&addr, sizeof(addr));
115
```

```
115
116         if(actionSendResult < 0)
117         {
118             perror("Ping Error");
119             exit(0);
120         }
121
122         // Prepare all the necessary variable to handle the response
123         unsigned int resAddressSize;
124         unsigned char res[30] = "";
125         struct sockaddr resAddress;
126
127         // Read the response from the remote host
128         int response = recvfrom(s, res, sizeof(res), 0, &resAddress,
129                                 &resAddressSize);
130
131         if( response > 0)
132         {
133             printf("Received %d bytes from %s : %s\n", response, ip_address, argv[1]);
134         }
135         else
136         {
137             perror("Response Error!!!!");
138             exit(0);
139         }
140     }
141     return 0;
142 }
```

Output



Problem Statement 4

Write a C program to find the host name 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:

surreal24_to_ours17_viewnorm.py | transform_util.py — commons | problem1.c | problem2.cpp | problem3.c | problem4.c

```c
/** @file problem4.c
 *  @brief Problem Statement 4 get the host name and the IP address of your computer.
 *         <command> <devicename>
 *  @author Ashutosh
 */

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


/** @brief Prolem Statement 4 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 host_name;

   // Get the host name
   host_name = gethostname(host, sizeof(host));
   if (host_name == -1) {
      perror("gethost_name");
      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
   printf("Host Name: %s\n", host);
   printf("IP Address is = %s\n" , inet_ntoa(( (struct sockaddr_in *)&ifr.ifr_addr )->sin_addr) );
   return 0;
}
```

Output

```
(base) djikstra@helios:~/Academic/CSN361/L1$ gcc problem4.c -o problem4
(base) djikstra@helios:~/Academic/CSN361/L1$ ./problem4 dev1
Host Name: helios
IP Address is = [redacted].127.0.0
(base) djikstra@helios:~/Academic/CSN361/L1$
```