**Дз 2**

1) Дан 32-разрядный IP-адрес, имеющий в восьмеричном представлении вид: [15502410544](tel:15502410544).

    Определить, к какому классу относится данный  IP-адрес, номер сети (в восьмеричном представлении), к которой относится  IP адрес.

1 5 5 0 2 4 1 0 5 4 4

001 101 101 000 010 100 001 000 101 100 100

Все число занимает 33 бита, следовательно убираем самый левый ноль.

Новый самый левый бит = 0, значит класс А

Номер сети — следующие 7 бит:

1 101 101 = 155 в восьмеричной системе.

2) Дан 32-разрядный IP-адрес, имеющий в шестнадцатеричном представлении вид: DF00BE20.

    Определить, к какому классу относится данный  IP-адрес, номер сети (в 16-ричном представлении), к которой относится  IP адрес, и десятичный номер хоста в сети.

D F 0 0 B E 2 0

1101 1111 0000 0000 1011 1110 0010 0000

110 — класс С

Номер сети:

111110000000010111110 = 1F00BE в шестрадцатеричной системе

Номер хоста:

00100000 = 23 в десятичной системе

3) Пусть процесс с  PID = 2021 породил два сыновних процесса с PID’ами 2022 и 2023:

 int main (int argc, char \*\* argv)   /\* PID=2021 \*/

   {

      If (fork()==0) { /\*PID = 2022 \*/

         printf("%d %d\n", getppid(), getpid());

         exit(0);

      }

      If (fork()==0) { /\*PID = 2023 \*/

        printf("%d\n", getpid());

        exit(0);

      }

      return 0;

   }

Считаем, что  printf работает атомарно (печатает сразу всё) и обращения ко всем системным вызовам успешно срабатывают. Перечислить ВСЕ возможные комбинации значений, которые могут быть выведены на стандартное устройство вывода в результате выполнения данной программы.

1.

2021 2022

2023

2.

1 2022

2023

3.

2023

2021 2022

4.

2023

1 2022

4) Пусть процесс с  PID = 3047 породил два сыновних процесса с PID’ами 3048 и 3049:

 int main (int argc, char \*\* argv)   /\* PID=3047 \*/

   {

      If (fork()==0) { /\*PID = 3048 \*/

         printf("%d\n",  getpid());

         exit(0);

      }

      wait(NULL);

      If (fork()==0) { /\*PID = 3049 \*/

        printf("%d %d\n", getpid(), getppid());

        exit(0);

      }

      return 0;

   }

Считаем, что  printf работает атомарно  и обращения ко всем системным вызовам успешно срабатывают. Перечислить ВСЕ возможные комбинации значений, которые могут быть выведены на стандартное устройство вывода в результате выполнения данной программы.

1.

3048

3049 1

2.

3048

3049 3047

5) Написать два варианта программы, которая исполняет следующую команду shell:

    gcc  1.c -o 1.exe -lm

Первый вариант программы на C:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

execl("/bin/gcc", "gcc", "1.c", "-o", “1.exe", "-Im", NULL;

return 0;

}

Второй вариант программы на C:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

pid\_t pid = fork();

if (pid == 0) {

execlp("gcc", "gcc", "1.c", "-o", "1.exe", "-lm", NULL);

} else if (pid > 0) {

wait(NULL);

} else {

printf("Ошибка при создании нового процесса\n");

}

return 0;

}