ACS. OS. HW9

Горбачева Маргарита Валерьевна | БПИ-245

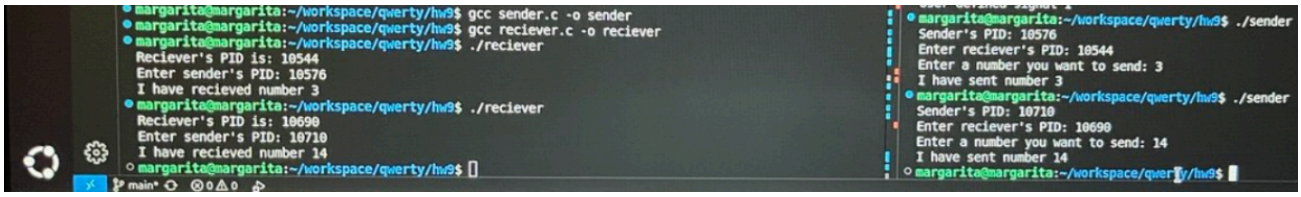
Задание:

Побитовая передача целого числа от одного процесса другому с помощью сигналов SIGUSR1, SIGUSR2. Необходимо написать две программы — передатчик и приемник, которые будут запускаться с разных терминалов. Каждая программа вначале печатает свой PID и запрашивает ввод с клавиатуры PID программы-собеседника (вводится пользователем). Порядок ввода PID не имеет значения. Программа—передатчик запрашивает с клавиатуры ввод целого десятичного числа (число может быть и отрицательным!). Введенное число побитово передается программе-приемнику с использованием пользовательских сигналов SIGUSR1 и SIGUSR2. Программа-приемник после завершения приема печатает принятое число в десятичной системе счисления. Примечание: Каждый бит (0 или 1) передается своим сигналом. Из-за ненадежности сигналов передача последовательности одинаковых битов может приводить к ошибкам. Поэтому каждый новый бит можно передавать только после подтверждения от приемника принятия предыдущего с помощью одного из пользовательских сигналов. Отсутствие подтверждения – грубая ошибка! Завершение передачи можно осуществить, например, передачей сигнала SIGINT.

Решение: решение данного домашнего задания лежит в папке GitHub репозитория по ссылке: https://github.com/Misss-Lacoste/ACS-OS/tree/main/hw9

NotaBene!! Прошу заметить, что комментарии в кодах написаны на английском языке, т.к. задание выполнялось на виртуальной машине линукса, которая не поддерживает русский язык.

Демонстрация работы программы:



Примечание: необходимо сплитануть терминал, скомпилировать два кода в одном из них, далее в первом терминале запустить код reciever'a, а во втором - код sender'a. Важно(!!!) именно в таком порядке, т.к. приёмник(receiver) должен быть готов принимать сигналы, последующие от отправителя(sender). В двух терминалах вводим PIDшники

sender’a и reciever’a. И в терминале, где запускали sender’a, вводим то число, котрое хотим перегнать receiver’у. Наслаждаемся общением терминалов 🙂

Как работает программа:

1. Рассмотрим код sender.c

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int reciever_pid;
int current_bit = 0;
int number_to_be_sent;

void send_bit(int bit) {
    if (bit == 0) {
        kill(reciever_pid, SIGUSR1); //send the receiver a signal to handle bit=0 and stop after this
    } else {
        kill(reciever_pid, SIGUSR2); //for bit=1
    }
}

void back_up_handler(int sig) {  //obrabotchik podtverzhdeniya from the receiver
    (void)sig;
    if(current_bit < 32) { //check that not all 32 bits are transmitted
        int bit = (number_to_be_sent >> current_bit) &1; //move our number right to current_bit positions and leave only the least bit(LSB)
        send_bit(bit);
        current_bit++;
    } else {
        kill(reciever_pid, SIGINT); //if all bits are transmitted, then send a signal to complete the transmission
        printf("I have sent number %d\n", number_to_be_sent);
        exit(0);
    }
}
int main() {
    printf("Sender's PID: %d\n", getpid());
    printf("Enter reciever's PID: ");
```

```
    scanf("%d", &reciever_pid);
    printf("Enter a number you want to send: ");
    scanf("%d", &number_to_be_sent);

    signal(SIGUSR1, back_up_handler);


    /*struct nichegonerabotaetuzhenemogy aaa;
    aaa.sa_handler = back_up_handler;
    aaa.sa_flags = 0;
    sigemptyset(&aaa_sa.mask);
    nichegonerabotaetuzhenemogy(SIGUSR1, &aaa, NULL);*/

    current_bit = 0;
    int first_bit = (number_to_be_sent >> current_bit) &1; //izvlekaem the
least significant bit
    send_bit(first_bit); //start the process: send the 1st bit
    current_bit++;

    while(1) {
        pause(); //wait for approval to continue
    }
    return 0;
}
```

2. Итак, мы написали два кода: для sender'а и receiver'а. Sender отправляет число receiver'у. Число, которое мы вводим, представлено в десятичной системе счисления, однако программа работает с двоичным представлением числа, которое мы ввели. Таким образом, sender передает receiver'у битики 32-разрядного двоичного числа, начиная с самого младшего бита, который находится с самого правого края записи числа в двоичном представлении, так называемый Least Significant Bit – LSB (привет лекциям по алгосам!). Если бит = 0, то вызывается SIGUSR1, иначе, если бит = 1, вызывается SIGUSR2.

3. Так, мы передали все побитово receiver'у. Теперь из этого фарша данных receiver должен воссоздать изначальное число. Стандартно умножаем бит (1 или 0) на 2(основание системы счисления) в показателе степени, в зависимости от того, в какой позиции стоит данный бит(начинаем справа с 0 индекса), и все суммируем (а теперь передаём привет любимому ЕГЭ по информатике!!). В коде receiver'а есть такая конструкция:

```
recieved_number |= (1u << bits); //here we set bit via creating mask
with 1 in 'bits' position
    bits++;
```

Это как раз та самая часть программы, когда приемник собирает число по крупицам обратно. То есть в переменную recieved_number будем последовательно складывать получаемые вычисляемые значения исходного числа. Оператор << означает сдвиг влево, то есть умножение на 2 в степени. То есть в переменной bits мы храним число битов, используемых в записи двоичного числа, таким образом, мы рассматриваем каждый бит(начиная с самого малого, который мы первым отправили receiver'у), такая запись эквивалентна следующей:
`(1 << bitsAmount)` = $1 \times 2^{bitsAmount} = 2^{bitsAmount}$.

Как-то так на человеческом языке:
```
1 << 0 = 1   //2⁰ = 1
1 << 1 = 2   //2¹ = 2
```

Ну и синтаксический сахар a |= b означает a = a | b. То есть используем побитовое или для установки битов.

4. Рассмотрим код receiver.c (полагаю, комментариев в коде достаточно, не хочется копипастить их в отчет).

```
#include <signal.h> //libraries for all used functions and syscalls
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> //for our favourite posix
#include <sys/types.h>

//struct signal {}

int sender_pid;
int recieved_number = 0;
int bits = 0; //number of recieved bits


void handle_sigusr1(int sig) { //aka obrabotchik of a signal(means bit=0 is
received)
    (void)sig; //ignore parametr of a signal, or the compiler will be angry
```

```
    bits++;
    kill(sender_pid, SIGUSR1); //send a msg to the sender that we have
received a bit=0
}

void handle_sigusr2(int sig) { //and now we are taking bit 1
    (void)sig;
    recieved_number |= (1u << bits); //here we set bit via creating mask
with 1 in 'bits' position
    bits++;
    kill(sender_pid, SIGUSR1); //again tell the sender that we have
received a bit=1
}
void handle_sigint(int sig) { //such obrabotchik means that transmission is
completed
    (void)sig;
    printf("I have recieved number %d\n", (int)recieved_number);
    exit(0);
}

int main() {
    printf("Reciever's PID is: %d\n", getpid());
    printf("Enter sender's PID: ");
    scanf("%d", &sender_pid);

    /*struct nichegonerabotaetuzhenemogy aaa;
    aaa.sa_flags = 0;
    sigemptyset(&aaa_sa.mask);

    aaa.sa_handler = handle_sigusr1;
    nichegonerabotaetuzhenemogy(SIGUSR1, &aaa, NULL);

    aaa.sa_handler = handle_sigusr2;
    nichegonerabotaetuzhenemogy(SIGUSR2, &aaa, NULL);

    printf("Vash zapros b obrabotke...\n");*/
    signal(SIGUSR1, handle_sigusr1);
    signal(SIGUSR2, handle_sigusr2);
    signal(SIGINT, handle_sigint);

    while(1) {
        pause(); //stops the program before getting another signal
    }
```

```
    return 0;
}
```

5. После того, как receiver собрал воедино изначально введенное число, он передает его нам на экран терминала(то есть предварительно получив SIGINT и вызвав handle_sigint(), печатает на экран число), затем выполняет exit(0) и программа завершается. В sender'е тоже выполняется exit(0) и программа уходит на покой.

P.S.: а это милая картиночка для ассистента :>