Process Execises

Exercise 1

Write a C program that forks a child, which forks a child. Every second, the grand-child sends a `SIGUSR1` to the the child (which is the father of the grand-child). When the child receives a `SIGUSR1`, it sends a `SIGUSR2` to the father.

```
        SIGUSR2       SIGUSR1
FATHER <------ CHILD <------ GRAND-CHILD
```

Solution:

```c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
void handler_child(int signum){
    printf("Child received SIGUSR1 and sends SIGUSR2\n");
    kill (getppid(), SIGUSR2);
}
void handler_father(int signum){
    printf("Father received SIGUSR2\n");
}
int main (){
    pid_t pid;
    struct sigaction action;
    pid = fork();
    if (pid!=0){ /* Father */
        action.sa_handler = handler_father;
        sigemptyset (&action.sa_mask);
        action.sa_flags = 0;
        sigaction (SIGUSR2, &action, NULL);
        while (1);
    } else{
        pid = fork();
        if (pid!=0){ /* Child */
            action.sa_handler = handler_child;
            sigemptyset (&action.sa_mask);
            action.sa_flags = 0;
            sigaction (SIGUSR1, &action, NULL);
            while (1);
        }
        else{ /* Grand-Child */
            while(1){
                sleep(1);
                kill (getppid(), SIGUSR1);
            }
        }
    }
}
```

Exercise 2

Write a C program that has two processes. The first process sends to the second the name of a file. The second process reads the name of the file and sends back to the first process its content.

Solution:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

void client(int readfd,int writefd){
   char buffer [1000];
   printf ("Pathname: ");

   scanf("%s", buffer);
   write(writefd, buffer, strlen(buffer));

   while (read(readfd,buffer,sizeof(buffer)) > 0)    printf("%s",buffer);
}

void server(int readfd,int writefd){
   FILE * fp;
   char buffer[1000];
   int n;

   n = read(readfd,buffer,sizeof(buffer));
   buffer[n]= 0;

   fp=fopen(buffer,"r");
   while(fgets(buffer,sizeof(buffer),fp)!=NULL)
       write(writefd,buffer,sizeof(buffer));
}

int main()
{
    int pfd1[2],pfd2[2], pid, status;

    pipe(pfd1); pipe(pfd2);
    pid=fork();
    if(pid>0)
    {
        close(pfd1[0]); close(pfd2[1]);
        client(pfd2[0], pfd1[1]);
        wait(&status);
        exit(0);
    }
    else
    {
        close(pfd1[1]); close(pfd2[0]);
        server(pfd1[0],pfd2[1]);
        exit(0);
    }
}
```

Exercise 3

Write a C program that has two processes and shared memory containing `N` floating point numbers. The child process fills the shared memory `float` vector, where each element `v[i]=10 * i`, then it terminates. The father process waits for the child to terminate, then it prints the vector.

Solution:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/mman.h>
#include <sys/wait.h>
#define N 10

int main (void)
{   int i;
    float* shared_memory;

    shared_memory=mmap(0, N*sizeof(float), PROT_READ | PROT_WRITE,
                       MAP_ANONYMOUS | MAP_SHARED, -1, 0);
    if (fork()){
        wait(NULL);
        for(i=0; i<N; i++)
            printf("shared_memory[%d]==%f\n", i, shared_memory[i]);
    }
    else    {
        for(i=0; i<N; i++)
            shared_memory[i]=i*10;
    }
    return 0;
}
```