Εργαστήριο 8
Ασκήσεις: Process Management and Signals

Για τη λύση του προβλήματος μπορούμε να χρησιμοποιήσουμε 3 τρόπους:

1. Χρήση FIFO pipes
2. Ο πατέρας και τα παιδιά αλλάζουν τον write descriptor σε κάτι που ξέρουν και οι δύο με τη χρήση της κλήσης συστήματος dup2().

Στα πλαίσια της λύσης που δίνεται χρησιμοποιούμε τον δεύτερο τρόπο, όπου ο πατέρας και τα παιδιά αλλάζουν τον write descriptor του pipe στο file descriptor 1.

```
//parent program
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#define pi
3.14159265358979323846264338327950288419716939937510582097494459230781640
628620899862803482534211706798214808651328

#define READ 0
#define WRITE 1

int pipe1[2], pipe2[2];
int status, pid1, pid2;
char response_pi1[70], response_pi2[70];

void Handler_C(int a)
{
     //signal(SIGINT, Handler_C);
     kill(pid1, SIGUSR1);
     kill(pid2, SIGUSR1);
     response_pi1[0]='\0';
     response_pi2[0]='\0';
     read(pipe1[READ], response_pi1, 70);
     read(pipe2[READ], response_pi2, 70);
     printf("The difference of pi and Compute_pi1 is %.30f\n",
atof(response_pi1)-pi);
     printf("The difference of pi and Compute_pi2 is %.30f\n",
atof(response_pi2)-pi);
}

void Handler_Z(int a)
{
     //signal(SIGTSTP, Handler_Z);
     kill(pid1, SIGUSR1);
     kill(pid2, SIGUSR1);
     response_pi1[0]='\0';
     response_pi2[0]='\0';
     read(pipe1[READ], response_pi1, 70);
     read(pipe2[READ], response_pi2, 70);
     if(atof(response_pi1)-pi < atof(response_pi2)-pi)
          printf("First approach is faster\n");
     else
          printf("Second approach is faster\n");
     kill(pid1, SIGKILL);
     kill(pid2, SIGKILL);
     exit(0);
```

```
}

int main ()
{
      int ret;
      if (pipe(pipe1)==-1)
      {
            printf("Error creating the first pipe \n");
            exit (1);
      }
      if (pipe(pipe2)==-1)
      {
            printf("Error creating the first pipe \n");
            exit (1);
      }
      pid1=fork();
      if (pid1==-1)
      {
            printf("Error creating the first child \n");
            exit (1);
      }
      if (pid1==0)
      {
            dup2(pipe1[1],1);
            close(pipe1[READ]);
            ret=execl("./compute_pi1", NULL);
            if (ret==-1)
            {
                  printf("Could not convert process one \n");
                  exit (1);
            }
      }
      else
      {
            pid2=fork();
            if (pid2==-1)
            {
                  printf("Error creating the second child \n");
                  exit(1);
            }
            if (pid2==0)
            {
                  dup2(pipe2[1],1);
                  close(pipe2[READ]);
                  ret=execl("./compute_pi2", NULL);
                  if (ret==-1)
                  {
                        printf ("Could not convert process number two \n
");
                        exit(1);
                  }
            }
            else
            {
                  printf ("The program started succesfully\n");
                  printf ("Press Ctrl^C to see the current values of pi and
Ctrl^Z to terminate\n\n");
                  close(pipe1[WRITE]);
                  close(pipe2[WRITE]);
```

```
                signal(SIGINT, Handler_C);
                signal(SIGTSTP, Handler_Z);
                wait(&status);
                wait(&status);
          }
     }
}

// compute_pi1
#include <stdio.h>
#include <signal.h>

float pi=0;

void Handler_Ch(int a)
{
     char current_pi1[70];
     signal(SIGUSR1, Handler_Ch);
     sprintf(current_pi1, "%.67f", pi);
     write(1, current_pi1, 70);
}

void compute_pi1()
{
     float counter=1;
     int flag=0;
     signal(SIGINT, SIG_IGN);
     signal(SIGTSTP, SIG_IGN);
     signal(SIGUSR1, Handler_Ch);
     while(1)
     {
          if (flag % 2==0)
               pi= pi+(4/counter);
          else
               pi= pi-(4/counter);
          flag++;
          counter+=2;
          sleep(1);
     }
}

int main()
{
     compute_pi1();
}


// compute_pi2
#include <stdio.h>
#include <signal.h>
#include <math.h>

float pi=0;

void Handler_Ch(int a)
{
     char current_pi1[70];
     signal(SIGUSR1, Handler_Ch);
     sprintf(current_pi1, "%.67f", sqrt(pi));
```

```
      write(1, current_pi1, 70);
}

void compute_pi1()
{
      float counter=1;
      signal(SIGINT, SIG_IGN);
      signal(SIGTSTP, SIG_IGN);
      signal(SIGUSR1, Handler_Ch);
      while(1)
      {
            pi = pi+(6/(counter*counter));
            counter++;
            sleep(1);
      }
}

int main()
{
      compute_pi1();
}
```