LAB 5.1

Bài 1

UnName

```c
//Bai1 UnName
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char **argv)
{
int fp1[2];

int pid;
//printf("%d",argc);
if (argc < 2)
{
printf("Doi so thieu.\n");
return -1;
}
if (pipe(fp1) == 0)
{
pid = fork();

if (pid < 0)
{
printf("Fork failed\n");
return -1;
}
else if (pid == 0)
{
close(fp1[1]);
char buffer[256];
while (read(fp1[0], &buffer, sizeof(buffer))>0)
{
printf("%s\n",buffer);
}
close(fp1[0]);
}
else
{
close(fp1[0]);
//o doc 1 viet
int i;
for (i = 1; i <argc; i++){
write(fp1[1],&argv[i],sizeof(&argv[i]));
}
//wait(NULL);
close(fp1[1]);
}
}
else
{
printf("Pipe failed\n");
return -2;
}
}

```

Name

```
//Name
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096
int main(int argc, char *argv[])
{
char s1[PIPE_BUF], s2[PIPE_BUF];
int childpid, readfd, writefd;
if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
{
printf("Fail to create FIFO 1. Aborted.\n");
```

```c
return -1;
}
if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
{
unlink(FIFO1);
printf("Fail to create FIFO 2. Aborted.\n");
return -1;
}
childpid = fork();
if (childpid == 0)
{ // child
if ((readfd = open(FIFO1, 0)) < 0)
perror("Child cannot open readFIFO.\n");
while(read(readfd, s2, PIPE_BUF)){
printf("%s\n", s2);
}
close(readfd);
return 1;
}
else if (childpid > 0)
{ // parent
if ((writefd = open(FIFO1, 1)) < 0)
perror("Parent cannot open writeFIFO.\n");
int i;
for (i = 1; i < argc; i++)
      {
      gets(s1);
      write(writefd, s1, PIPE_BUF);
      }
      while (wait((int *)0) != childpid);
      close(writefd);
      if (unlink(FIFO1) < 0)
      perror("Cannot remove FIFO1.\n");
      return 1;
      }
      else
      {
      printf("Fork failed\n");
      return -1;
      }
      }
```

Bài 2

UnName

```c
//UnName
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[])
{
int fp1[2], fp2[2];
int pid;
if (argc < 2)
{
printf("Doi so thieu.\n");
return -1;
}
if (pipe(fp1) == 0)
{
pid = fork();

if (pid < 0)
{
printf("Fork failed\n");
return -1;
}
else if (pid == 0)
{
int buffer;
close(fp1[1]);
read(fp1[0], &buffer, sizeof(buffer));
int cnt = 1, i;
for (i = 1; i <= buffer; i++)
{
cnt *= i;
}
printf("%d!=%d\n", buffer, cnt);
close(fp1[0]);
}
else
{
```

```
close(fp1[0]);
// o doc 1 viet
int tmp = atoi(argv[1]);
write(fp1[1], &tmp, sizeof(tmp));
close(fp1[1]);
}
}
else
{
printf("Pipe failed\n");
return -2;
}
}
```

Name

```
//Name
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096
int main(int argc, char *argv[])
{
char s1[PIPE_BUF], s2[PIPE_BUF];
int childpid, readfd, writefd;
if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
{
printf("Fail to create FIFO 1. Aborted.\n");
return -1;
}
if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
{
unlink(FIFO1);
printf("Fail to create FIFO 2. Aborted.\n");
```

```c
return -1;
}
childpid = fork();
if (childpid == 0)
{ // child
if ((readfd = open(FIFO1, 0)) < 0)
perror("Child cannot open readFIFO.\n");
fflush(stdin);
read(readfd, s2, PIPE_BUF);
int cnt = 1;
int i;
for (i = 1; i <= atoi(s2); i++)
{
cnt *= i;
}
printf("%d!=%d\n", atoi(s2), cnt);
close(readfd);
return 1;
}
else if (childpid > 0)
{ // parent
if ((writefd = open(FIFO1, 1)) < 0)
perror("Parent cannot open writeFIFO.\n");
fflush(stdin);
scanf("%s",s1);
write(writefd, s1,strlen(s1));
while (wait((int *)0) != childpid)
;
close(writefd);
if (unlink(FIFO1) < 0)
perror("Cannot remove FIFO1.\n");
return 1;
}
else
{
printf("Fork failed\n");
return -1;
}
}
```

Bài 3

Name

```c
//Bai3 UnName
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[])
{
int fp1[2], fp2[2];
int pid;
char arr[argc + 1];
if (argc < 2)
{
printf("Doi so thieu.\n");
return -1;
}
if (pipe(fp1) == 0)
{
pid = fork();

if (pid < 0)
{
printf("Fork failed\n");
return -1;
}
else if (pid == 0)
{
char buffer[100];
close(fp1[1]);
char arr[100];
int cnt = 0;

fflush(stdin);
//read(fp1[0], &buffer, sizeof(buffer)) ;
//printf("%s",buffer);
//scanf("%s%s%s",arr[0],arr[1]),arr[2]);
while (read(fp1[0], &buffer, sizeof(buffer))!= 0)
{
strcpy(arr[cnt],buffer);
cnt++;
//printf("%s",buffer);
}
/*
  switch(arr[2]){
  case '+':
  printf("%d+%d=%d\n",atoi(arr[0]),atoi(arr[1]),atoi(arr[0]) + atoi(arr[1]));
  break;
  case '-':
  printf("%d-%d=%d\n",atoi(arr[0]),atoi(arr[1]),atoi(arr[0]) - atoi(arr[1]));
  break;
  case '*':
  printf("%d*%d=%d\n",atoi(arr[0]),atoi(arr[1]),atoi(arr[0]) * atoi(arr[1]));
  break;
  case '/':
  if(atoi(arr[1])==0){
  printf("Khong chia duoc cho 0\n");
  }else{
  printf("%d/%d=%d\n",atoi(arr[0]),atoi(arr[1]),atoi(arr[0]) / atoi(arr[1]));
  }
  break;
  default:
  printf("Khong co toan tu\n");
  break;
  }
*/
close(fp1[0]);
}
else
{
close(fp1[0]);
int i;
fflush(stdin);
for (i = 1; i < argc; i++)
{
write(fp1[1], argv[i], strlen(argv[i]));
}
close(fp1[1]);
}
}
else
{
printf("Pipe failed\n");
return -2;
}
```

```
}
```

Name

```
//Name
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096
int main(int argc, char *argv[])
{
char s1[PIPE_BUF], s2[PIPE_BUF];
int childpid, readfd, writefd;
if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
{
printf("Fail to create FIFO 1. Aborted.\n");
return -1;
}
if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
{
unlink(FIFO1);
printf("Fail to create FIFO 2. Aborted.\n");
return -1;
}
childpid = fork();
if (childpid == 0)
{ // child
if ((readfd = open(FIFO1, 0)) < 0)
perror("Child cannot open readFIFO.\n");
char arr[4];
int cnt=0;
fflush(stdin);
while (read(readfd, s1, PIPE_BUF) > 0){
arr[cnt++] = s1[0];
```

```c
}
printf("%s",s1[0]);
close(readfd);
return 1;
}
else if (childpid > 0)
{ // parent
if ((writefd = open(FIFO1, 1)) < 0)
perror("Parent cannot open writeFIFO.\n");
int i;
for (i = 0; i < argc; i++)
{
fflush(stdin);
scanf("%s",s1);
write(writefd, s1, PIPE_BUF);
}
while (wait((int *)0) != childpid)
;
close(writefd);
if (unlink(FIFO1) < 0)
perror("Cannot remove FIFO1.\n");
return 1;
}
else
{
printf("Fork failed\n");
return -1;
}
}
```