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");
```

```
75. return -1;
76. }
77. if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
78. {
79. unlink(FIFO1);
80. printf("Fail to create FIFO 2. Aborted.\n");
81. return -1;
82. }
83. childpid = fork();
84. if (childpid == 0)
85. { // child
86. if ((readfd = open(FIFO1, 0)) < 0)
87. perror("Child cannot open readFIFO.\n");
88. while(read(readfd, s2, PIPE_BUF)){
89. printf("%s\n", s2);
90. }
91. close(readfd);
92. return 1;
93. }
94. else if (childpid > 0)
95. { // parent
96. if ((writefd = open(FIFO1, 1)) < 0)
97. perror("Parent cannot open writeFIFO.\n");
98. int i;
99. for (i = 1; i < argc; i++)
100.       {
101.       gets(s1);
102.       write(writefd, s1, PIPE_BUF);
103.       }
104.       while (wait((int *)0) != childpid);
105.       close(writefd);
106.       if (unlink(FIFO1) < 0)
107.       perror("Cannot remove FIFO1.\n");
108.       return 1;
109.       }
110.       else
111.       {
112.       printf("Fork failed\n");
113.       return -1;
114.       }
115.       }
```

Bài 2

UnName

```
1. //UnName
2. #include <stdio.h>
3. #include <unistd.h>
4. #include <string.h>
5.
6. int main(int argc, char *argv[])
7. {
8. int fp1[2], fp2[2];
9. int pid;
10. if (argc < 2)
11. {
12. printf("Doi so thieu.\n");
13. return -1;
14. }
15. if (pipe(fp1) == 0)
16. {
17. pid = fork();
18.
19. if (pid < 0)
20. {
21. printf("Fork failed\n");
22. return -1;
23. }
24. else if (pid == 0)
25. {
26. int buffer;
27. close(fp1[1]);
28. read(fp1[0], &buffer, sizeof(buffer));
29. int cnt = 1, i;
30. for (i = 1; i <= buffer; i++)
31. {
32. cnt *= i;
33. }
34. printf("%d!=%d\n", buffer, cnt);
35. close(fp1[0]);
36. }
37. else
38. {
```

```
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");
```

```
27. return -1;
28. }
29. childpid = fork();
30. if (childpid == 0)
31. { // child
32. if ((readfd = open(FIFO1, 0)) < 0)
33. perror("Child cannot open readFIFO.\n");
34. fflush(stdin);
35. read(readfd, s2, PIPE_BUF);
36. int cnt = 1;
37. int i;
38. for (i = 1; i <= atoi(s2); i++)
39. {
40. cnt *= i;
41. }
42. printf("%d!=%d\n", atoi(s2), cnt);
43. close(readfd);
44. return 1;
45. }
46. else if (childpid > 0)
47. { // parent
48. if ((writefd = open(FIFO1, 1)) < 0)
49. perror("Parent cannot open writeFIFO.\n");
50. fflush(stdin);
51. scanf("%s",s1);
52. write(writefd, s1,strlen(s1));
53. while (wait((int *)0) != childpid)
54. ;
55. close(writefd);
56. if (unlink(FIFO1) < 0)
57. perror("Cannot remove FIFO1.\n");
58. return 1;
59. }
60. else
61. {
62. printf("Fork failed\n");
63. return -1;
64. }
65. }
```

Bài 3

Name

```
1. //Bai3 UnName
2. #include <stdio.h>
3. #include <unistd.h>
4. #include <string.h>
5.
6. int main(int argc, char *argv[])
7. {
8. int fp1[2], fp2[2];
9. int pid;
10. char arr[argc + 1];
11. if (argc < 2)
12. {
13. printf("Doi so thieu.\n");
14. return -1;
15. }
16. if (pipe(fp1) == 0)
17. {
18. pid = fork();
19.
20. if (pid < 0)
21. {
22. printf("Fork failed\n");
23. return -1;
24. }
25. else if (pid == 0)
26. {
27. char buffer[100];
28. close(fp1[1]);
29. char arr[100];
30. int cnt = 0;
31.
32. fflush(stdin);
33. //read(fp1[0], &buffer, sizeof(buffer)) ;
34. //printf("%s",buffer);
35. //scanf("%s%s%s",arr[0],arr[1]),arr[2]);
36. while (read(fp1[0], &buffer, sizeof(buffer))!= 0)
37. {
38. strcpy(arr[cnt],buffer);
39. cnt++;
40. //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;
}
```

```
84. }
```

Name

```
1. //Name
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <unistd.h>
5. #include <string.h>
6. #include <sys/types.h>
7. #include <sys/stat.h>
8. #include <sys/errno.h>
9. #define FIFO1 "/tmp/ff.1"
10. #define FIFO2 "/tmp/ff.2"
11. #define PM 0666
12. extern int errno;
13. #define PIPE_BUF 4096
14. int main(int argc, char *argv[])
15. {
16. char s1[PIPE_BUF], s2[PIPE_BUF];
17. int childpid, readfd, writefd;
18. if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
19. {
20. printf("Fail to create FIFO 1. Aborted.\n");
21. return -1;
22. }
23. if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
24. {
25. unlink(FIFO1);
26. printf("Fail to create FIFO 2. Aborted.\n");
27. return -1;
28. }
29. childpid = fork();
30. if (childpid == 0)
31. { // child
32. if ((readfd = open(FIFO1, 0)) < 0)
33. perror("Child cannot open readFIFO.\n");
34. char arr[4];
35. int cnt=0;
36. fflush(stdin);
37. while (read(readfd, s1, PIPE_BUF) > 0){
38. arr[cnt++] = s1[0];
```

```
39. }
40. printf("%s",s1[0]);
41. close(readfd);
42. return 1;
43. }
44. else if (childpid > 0)
45. { // parent
46. if ((writefd = open(FIFO1, 1)) < 0)
47. perror("Parent cannot open writeFIFO.\n");
48. int i;
49. for (i = 0; i < argc; i++)
50. {
51. fflush(stdin);
52. scanf("%s",s1);
53. write(writefd, s1, PIPE_BUF);
54. }
55. while (wait((int *)0) != childpid)
56. ;
57. close(writefd);
58. if (unlink(FIFO1) < 0)
59. perror("Cannot remove FIFO1.\n");
60. return 1;
61. }
62. else
63. {
64. printf("Fork failed\n");
65. return -1;
66. }
67. }
```