
实验二 进程和进程通信

操作系统实验报告

郑宇森

520021911173

电子信息与电气工程学院

zys0794@sjtu.edu.cn

2022 年 11 月 27 日

目录

1 实验要求	2
2 父子进程协作实验	2
2.1 程序逻辑	2
2.2 程序源码	3
2.3 程序输出	3
3 消息通信实验	4
3.1 程序逻辑	4
3.2 程序源码	4
3.3 程序输出	6
4 共享内存实验	7
4.1 程序逻辑	7
4.2 程序源码	7
4.3 程序输出	9
5 实验总结	9
A utils.h 文件	10
B Makefile 文件	12

1 实验要求

- Part1：自己设计一个程序，该程序创建一个子进程，使父子进程合作，协调地完成某一功能。要求在该程序中还要使用进程的睡眠、进程图象改换、父进程等待子进程终止、信号的设置与传送（包括信号处理程序）、子进程的终止等有关进程的系统调用。
- Part 2：分别利用 UNIX 的消息通信机制、共享内存机制（用信号灯实施进程间的同步和互斥）实现两个进程间的数据通信。具体的通信数据可从一个文件读出，接收方进程可将收到的数据写入一个新文件，以便能判断数据传送的正确性。

2 父子进程协作实验

该部分的实验程序在 `procfork.c` 中。可通过 `make procfrok` 编译。

2.1 程序逻辑

本实验程序实现了以下功能，包含了 Part1 的所有测试要求。

1. 父进程创建一个子进程
2. 子进程睡眠，等待父进程唤醒
3. 父进程分别将字符串 `I love SJTU!` 与 `I love Operation System!` 写入文件 `file1.txt` 与 `file2.txt`
4. 父进程向子进程发送信号，唤醒睡眠的子进程
5. 父进程等待子进程终止
6. 子进程执行信号处理程序。
7. 子进程图像改换为 `shell` 命令 `cat`
8. 子进程读取和连接文件 `file1.txt` 与 `file2.txt`，并将其内容写入到标准输出
9. 子进程终止
10. 父进程打印返回状态 `status` 并结束

2.2 程序源码

```
1 #include "utils.h"
2 void func() { printf("Aha! I am alive!\n"); }
3 int main() {
4     int status;
5     pid_t pid;
6     signal(SIGUSR1, func);
7
8     if ((pid = fork())) {
9         FILE *f1 = fopen("file1.txt", "w");
10        FILE *f2 = fopen("file2.txt", "w");
11        if (f1 == NULL) {
12            printf("Error opening file1.txt!\n");
13            exit(1);
14        }
15        if (f2 == NULL) {
16            printf("Error opening file2.txt!\n");
17            exit(1);
18        }
19        const char *text1 = "I love SJTU!";
20        const char *text2 = "I love Operation Systems!";
21        fprintf(f1, "Text1: %s\n", text1);
22        fprintf(f2, "Text2: %s\n", text2);
23        fclose(f1);
24        fclose(f2);
25        printf(ANSI_COLOR_GREEN "Parent: " ANSI_COLOR_RESET "Signal send.\n");
26        kill(pid, SIGUSR1);
27        wait(&status);
28        printf(ANSI_COLOR_GREEN "Parent: " ANSI_COLOR_RESET
29              "Process finished, status = %d\n",
30              status);
31    } else {
32        sleep(10);
33        printf(ANSI_COLOR_GREEN "Child: " ANSI_COLOR_RESET "Signal received\n");
34        execl("/bin/cat", "cat", "./file1.txt", "./file2.txt", (char *)0);
35        printf(ANSI_COLOR_RED "execl error!\n" ANSI_COLOR_RESET);
36        exit(2);
37    }
38 }
```

2.3 程序输出

程序输出见图 1，可见其完成了所有预期功能。

```
● > make procfork
gcc -Wall -O3 -o procfork procfork.c

lab/lab2 on 1 NIS3325 via C v9.4.0-gcc took 2s
● > ./procfork
Parent: Signal send.
Aha! I am alive!
Child: Signal received.
Text1: I love SJTU!
Text2: I love Operation Systems!
Parent: Process finished, status = 0
```

图 1: 父子进程协作实验输出

3 消息通信实验

该部分的实验程序在 `client.c` 与 `server.c` 中。可通过 `make client server` 编译。

3.1 程序逻辑

本程序旨在利用消息通信机制，发送方从一个文件读出通信数据，写入消息通信机构管理的消息缓冲区；接收方从消息缓冲区中读出通信数据，将通信数据写入另一个文件中。程序逻辑如下。

1. 客户端进程每次读取文件 `in.txt` 中的一行数据（该文件的内容为狄兰·托马斯创作的诗歌 *Do not go gentle into that good night*）
2. 客户端调用 `msgsnd`，向服务器进程发送 `mtype` 为 1、`mtext` 为读取的数据、`msize` 为数据长度、`mpid` 为进程标识数的消息
3. 服务器进程调用 `msgrcv` 接收到客户端进程发送的消息
4. 服务器进程将消息的 `mtext` 字段的内容写入文件 `out.txt` 中
5. 服务器进程向客户端进程回发 `mtype` 为客户端进程 `pid` 的消息
6. 客户端进程接收服务器进程回发的消息
7. 循环步骤 1-6，直到对文件 `in.txt` 的读取和对文件 `out.txt` 的写入完成

3.2 程序源码

本实验采用的消息数据结构见附录 A 中 `struct msgtype` 结构体部分。

客户端的程序如下

```
1 #include "utils.h"
2 int main() {
3     struct msgtype buf;
4     pid_t qid, pid;
5     FILE *fin = fopen("in.txt", "r+");
6     if (fin == NULL) {
7         perror("fopen");
8         exit(1);
9     }
10    qid = msgget(MSGKEY, IPC_CREAT | 0666);
11    while (fgets(buf.mtext, BUFFER_SIZE, fin) > 0) {
12        buf.mpid = pid = getpid();
13        buf.mtype = MSGTYPE;
14        buf.msize = strlen(buf.mtext);
15        msgsnd(qid, &buf, sizeof(buf.mtext), 0);
16        printf(ANSI_COLOR_GREEN "Client: " ANSI_COLOR_RESET
17              "Send a message to server\nmpid: "
18              ANSI_COLOR_BLUE
19              "%d" ANSI_COLOR_RESET ", mtext: "
20              ANSI_COLOR_BLUE
21              "%s" ANSI_COLOR_RESET,
22              buf.mpid, buf.mtext);
23        msgrcv(qid, &buf, BUFFER_SIZE, pid, MSG_NOERROR);
24        printf(ANSI_COLOR_GREEN
25              "Client: " ANSI_COLOR_RESET
26              "Received a message from server, type: " ANSI_COLOR_BLUE
27              "%ld" ANSI_COLOR_RESET "\n",
28              buf.mtype);
29    }
30    buf.mpid = pid = getpid();
31    buf.mtype = MSGTYPE;
32    buf.msize = 0;
33    msgsnd(qid, &buf, BUFFER_SIZE, 0);
34    fclose(fin);
35    return 0;
36 }
```

服务器端的程序如下

```
1 #include "utils.h"
2 int main() {
3     struct msgtype buf;
4     pid_t qid;
5     FILE *fout = fopen("out.txt", "w+");
6     if (fout == NULL) {
7         perror("fopen");
8         return -1;
9     }
10    if ((qid = msgget(MSGKEY, IPC_CREAT | 0666)) == -1) {
11        return -1;
12    }
```

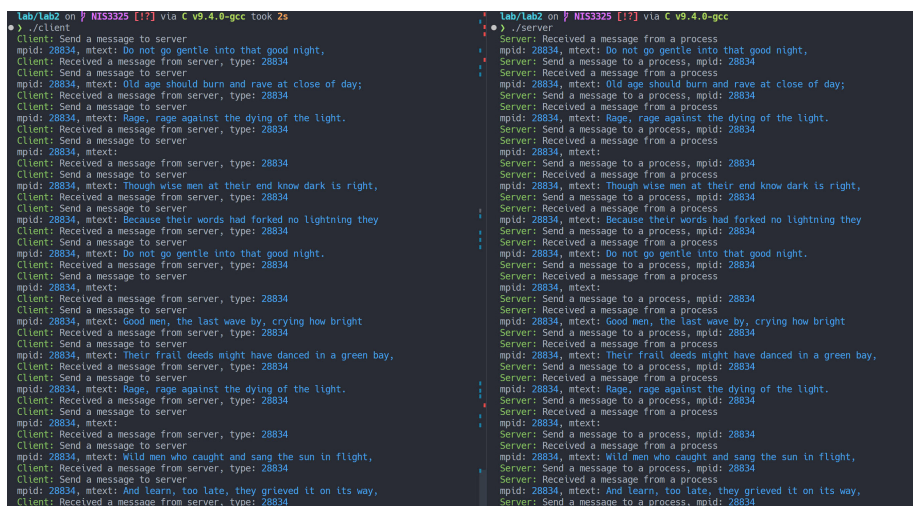
```

12 }
13 while (1) {
14     msgrcv(qid, &buf, BUFFER_SIZE, MSGTYPE, MSG_NOERROR);
15     if (buf.msize == 0) {
16         break;
17     }
18     fputs(buf.mtext, fout);
19     printf(ANSI_COLOR_GREEN
20         "Server: " ANSI_COLOR_RESET
21         "Received a message from a process\nmpid: " ANSI_COLOR_BLUE
22         "%d" ANSI_COLOR_RESET ", mtext: " ANSI_COLOR_BLUE
23         "%s" ANSI_COLOR_RESET ,
24         buf.mpid, buf.mtext);
25     buf.mtype = buf.mpid;
26     msgsnd(qid, &buf, sizeof(buf.mtext), 0);
27     printf(ANSI_COLOR_GREEN
28         "Server: " ANSI_COLOR_RESET
29         "Send a message to a process, mpid: " ANSI_COLOR_BLUE
30         "%d" ANSI_COLOR_RESET "\n",
31         buf.mpid);
32 }
33 fclose(fout);
34 return 0;
35 }

```

3.3 程序输出

程序输出如图 2, 3 所示。可见客户进程和服务器进程通过使用消息通信机制, 协同实现了预期功能, 将 in.txt 中的内容写入 out.txt 中。



```

Lab/Lab2 on [?] NIS3325 [17] via C v9.4.0-gcc took 2s
./client
Client: Send a message to server
mpid: 28834, mtext: Do not go gentle into that good night,
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Old age should burn and rave at close of day;
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Rage, rage against the dying of the light.
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext:
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Though wise men at their end know dark is right,
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Because their words had forked no lightning they
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Do not go gentle into that good night.
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext:
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Good men, the last wave by, crying how bright
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Their frail deeds might have danced in a green bay,
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Rage, rage against the dying of the light.
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext:
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: Wild men who caught and sang the sun in flight,
Client: Received a message from server, type: 28834
Client: Send a message to server
mpid: 28834, mtext: And learn, too late, they grieved it on its way,
Client: Received a message from server, type: 28834

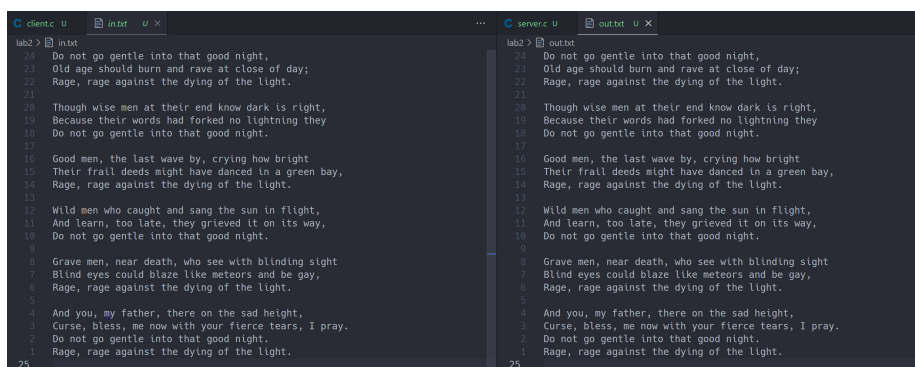
```

```

Lab/Lab2 on [?] NIS3325 [17] via C v9.4.0-gcc
./server
Server: Received a message from a process
mpid: 28834, mtext: Do not go gentle into that good night,
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Old age should burn and rave at close of day;
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Rage, rage against the dying of the light.
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext:
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Though wise men at their end know dark is right,
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Because their words had forked no lightning they
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Do not go gentle into that good night.
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext:
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Good men, the last wave by, crying how bright
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Their frail deeds might have danced in a green bay,
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Rage, rage against the dying of the light.
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext:
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: Wild men who caught and sang the sun in flight,
Server: Send a message to a process, mpid: 28834
Server: Received a message from a process
mpid: 28834, mtext: And learn, too late, they grieved it on its way,
Server: Send a message to a process, mpid: 28834

```

图 2: 消息通信实验 CLI 输出



```
client.c U  in.txt U X
lab2> in.txt
24 Do not go gentle into that good night,
23 Old age should burn and rave at close of day;
22 Rage, rage against the dying of the light.
21
20 Though wise men at their end know dark is right,
19 Because their words had forked no lightning they
18 Do not go gentle into that good night.
17
16 Good men, the last wave by, crying how bright
15 Their frail deeds might have danced in a green bay,
14 Rage, rage against the dying of the light.
13
12 Wild men who caught and sang the sun in flight,
11 And learn, too late, they grieved it on its way,
10 Do not go gentle into that good night.
9
8 Grave men, near death, who see with blinding sight
7 Blind eyes could blaze like meteors and be gay,
6 Rage, rage against the dying of the light.
5
4 And you, my father, there on the sad height,
3 Curse, bless, me now with your fierce tears, I pray.
2 Do not go gentle into that good night.
1 Rage, rage against the dying of the light.
25

server.c U  out.txt U X
lab2> out.txt
24 Do not go gentle into that good night,
23 Old age should burn and rave at close of day;
22 Rage, rage against the dying of the light.
21
20 Though wise men at their end know dark is right,
19 Because their words had forked no lightning they
18 Do not go gentle into that good night.
17
16 Good men, the last wave by, crying how bright
15 Their frail deeds might have danced in a green bay,
14 Rage, rage against the dying of the light.
13
12 Wild men who caught and sang the sun in flight,
11 And learn, too late, they grieved it on its way,
10 Do not go gentle into that good night.
9
8 Grave men, near death, who see with blinding sight
7 Blind eyes could blaze like meteors and be gay,
6 Rage, rage against the dying of the light.
5
4 And you, my father, there on the sad height,
3 Curse, bless, me now with your fierce tears, I pray.
2 Do not go gentle into that good night.
1 Rage, rage against the dying of the light.
25
```

图 3: 消息通信实验文件读写结果

4 共享内存实验

该部分的实验程序在 `procshm.c` 中。可通过 `make procshm` 编译。

4.1 程序逻辑

本程序旨在利用共享内存机制，父进程从一个文件读出通信数据，写入共享内存，子进程从共享内存中读出通信数据，将数据写入另一个文件中。父子进程的同步与互斥通过信号灯机制实现。程序逻辑如下。

1. 创建两个信号量 `sid1` 与 `sid2`，分别代表子进程是否读出数据与父进程是否写入数据。赋初值为 `sid1=1`, `sid2=0`
2. 父进程执行 `sid1←sid1-1`，若结果大于等于 0，说明子进程已从共享内存读出数据，则父进程可以从文件 `in.txt` 读出新的一行数据，写入共享内存中。写入操作结束后，执行 `sid2←sid2+1`
3. 子进程执行 `sid2←sid2-1`，若结果大于等于 0，说明父进程已从共享内存写入数据，则子进程可以从共享内存中读出新的一行数据，写入文件 `out.txt` 中。写入结束后，执行 `sid1←sid1+1`
4. 循环步骤 2-3，直到父进程读取到文件末尾。这时父进程通过 `kill` 函数向子进程发送信号，告知子进程通信结束。之后父子进程均关闭文件流。

4.2 程序源码

本实验使用的函数 `creatsem(key_t key)`, `semcall(int semid, int op)`, `P(int semid)`, `Q(int semid)` 见附录 A。

父子进程通过共享内存机制通信程序：

```
1 #include "utils.h"
2 extern int creatsem();
3 extern void P(), V();
4 static int flag = 1;
5 void func() { flag = 0; }
6 int main() {
7     time_t t;
8     srand((unsigned)time(&t));
9     FILE *fin = fopen("in.txt", "r+");
10    FILE *fout = fopen("out.txt", "w+");
11    char *segaddr;
12    int segid, sid1, sid2;
13    pid_t pid;
14    int status;
15    if ((segid = shmget(SHMKEY, SHM_SIZE, IPC_CREAT | 0666)) == -1) {
16        perror("shmget");
17    }
18    segaddr = shmat(segid, 0, 0);
19    sid1 = creatsem(SEMKEY1);
20    sid2 = creatsem(SEMKEY2);
21    P(sid2);
22    signal(SIGUSR1, func);
23    if (!(pid = fork())) {
24        while (1) {
25            P(sid2);
26            if (flag) {
27                fputs(segaddr, fout);
28                printf(ANSI_COLOR_GREEN "Child" ANSI_COLOR_RESET
29                    " Received a message: %s",
30                        segaddr);
31                V(sid1);
32            } else {
33                fclose(fout);
34                V(sid1);
35                break;
36            }
37        }
38    } else {
39        while (1) {
40            P(sid1);
41            if (fgets(segaddr, BUFFER_SIZE, fin)) {
42                printf(ANSI_COLOR_GREEN "Parent" ANSI_COLOR_RESET " Send a
43                    message: %s",
44                        segaddr);
45                V(sid2);
46            } else {
47                fclose(fin);
48                V(sid2);
49                kill(pid, SIGUSR1);
50            }
51        }
52    }
```



```

49     wait(&status);
50     break;
51 }
52 }
53 }
54 return 0;
55 }

```

4.3 程序输出

程序输出如图 4, 5所示。可见父子进程通过使用共享内存段, 协同实现了预期功能, 将 `in.txt` 中的内容写入 `out.txt` 中。从图 4也可看出, 通过信号灯机制, 父子进程依次按序写入/读出共享内存的内存, 实现了父子进程之间的同步。

```

Lab/lab2 on / NIS3325 via C v9.4.0-gcc took 2s
./proshm
Parent Send a message: Do not go gentle into that good night,
Child Received a message: Do not go gentle into that good night,
Parent Send a message: Old age should burn and rave at close of day;
Child Received a message: Old age should burn and rave at close of day;
Parent Send a message: Rage, rage against the dying of the light.
Child Received a message: Rage, rage against the dying of the light.
Parent Send a message:
Child Received a message:
Parent Send a message: Though wise men at their end know dark is right,
Child Received a message: Though wise men at their end know dark is right,
Parent Send a message: Because their words had forked no lightning they
Child Received a message: Because their words had forked no lightning they
Parent Send a message: Do not go gentle into that good night.
Child Received a message: Do not go gentle into that good night.
Parent Send a message:
Child Received a message:
Parent Send a message: Good men, the last wave by, crying how bright
Child Received a message: Good men, the last wave by, crying how bright
Parent Send a message: Their frail deeds might have danced in a green bay,
Child Received a message: Their frail deeds might have danced in a green bay,
Parent Send a message: Rage, rage against the dying of the light.
Child Received a message: Rage, rage against the dying of the light.
Parent Send a message:
Child Received a message:
Parent Send a message: Wild men who caught and sang the sun in flight,
Child Received a message: Wild men who caught and sang the sun in flight,
Parent Send a message: And learn, too late, they grieved it on its way,
Child Received a message: And learn, too late, they grieved it on its way,
Parent Send a message: Do not go gentle into that good night.
Child Received a message: Do not go gentle into that good night.
Parent Send a message:
Child Received a message:
Parent Send a message: Grave men, near death, who see with blinding sight
Child Received a message: Grave men, near death, who see with blinding sight
Parent Send a message: Blind eyes could blaze like meteors and be gay,
Child Received a message: Blind eyes could blaze like meteors and be gay,
Parent Send a message: Rage, rage against the dying of the light.
Child Received a message: Rage, rage against the dying of the light.
Parent Send a message:
Child Received a message:
Parent Send a message: And you, my father, there on the sad height,
Child Received a message: And you, my father, there on the sad height,
Parent Send a message: Curse, bless, me now with your fierce tears, I pray,
Child Received a message: Curse, bless, me now with your fierce tears, I pray,
Parent Send a message: Do not go gentle into that good night.
Child Received a message: Do not go gentle into that good night.
Parent Send a message: Rage, rage against the dying of the light.
Child Received a message: Rage, rage against the dying of the light.

```

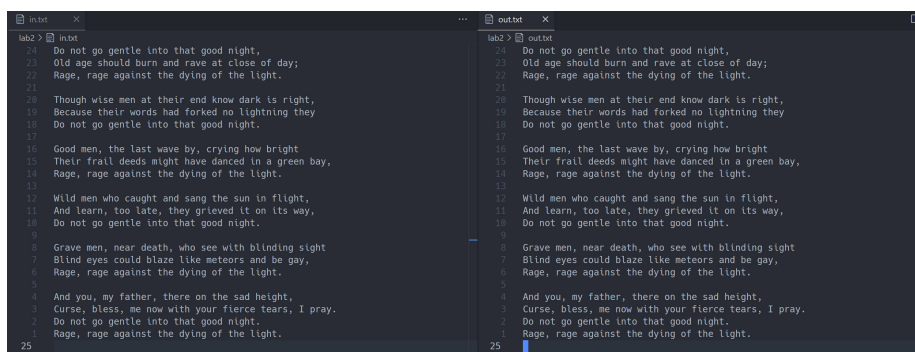
图 4: 共享内存实验 CLI 输出

5 实验总结

在实验中我遇到的问题主要如下:

进程通信中, 接收方进程在 `while(1)` 循环内不断检查是否收到消息。在发送方结束发送后, 接收方进程如何判断通信结束, 跳出 `while(1)` 循环?

通过实验尝试, 我采用了以下方法解决:



```

lab2> in.txt
24 Do not go gentle into that good night,
23 Old age should burn and rave at close of day;
22 Rage, rage against the dying of the light.
21
20 Though wise men at their end know dark is right,
19 Because their words had forked no lightning they
18 Do not go gentle into that good night.
17
16 Good men, the last wave by, crying how bright
15 Their frail deeds might have danced in a green bay,
14 Rage, rage against the dying of the light.
13
12 Wild men who caught and sang the sun in flight,
11 And learn, too late, they grieved it on its way,
10 Do not go gentle into that good night.
9
8 Grave men, near death, who see with blinding sight
7 Blind eyes could blaze like meteors and be gay,
6 Rage, rage against the dying of the light.
5
4 And you, my father, there on the sad height,
3 Curse, bless, me now with your fierce tears, I pray.
2 Do not go gentle into that good night.
1 Rage, rage against the dying of the light.
25

lab2> out.txt
24 Do not go gentle into that good night,
23 Old age should burn and rave at close of day;
22 Rage, rage against the dying of the light.
21
20 Though wise men at their end know dark is right,
19 Because their words had forked no lightning they
18 Do not go gentle into that good night.
17
16 Good men, the last wave by, crying how bright
15 Their frail deeds might have danced in a green bay,
14 Rage, rage against the dying of the light.
13
12 Wild men who caught and sang the sun in flight,
11 And learn, too late, they grieved it on its way,
10 Do not go gentle into that good night.
9
8 Grave men, near death, who see with blinding sight
7 Blind eyes could blaze like meteors and be gay,
6 Rage, rage against the dying of the light.
5
4 And you, my father, there on the sad height,
3 Curse, bless, me now with your fierce tears, I pray.
2 Do not go gentle into that good night.
1 Rage, rage against the dying of the light.
25

```

图 5: 共享内存实验文件读写结果

消息通信实验中,客户端进程在结束发送前,发送一条特殊的 `msize=0` 的消息,当服务器进程接收到该条消息后,便知晓消息通信结束,通过 `break` 跳出 `while(1)` 循环。

共享内存实验中,维护一个初值为 1 的全局变量 `flag`,当调用信号处理函数 `func` 时会将 `flag` 置零。子进程每次执行时都检查 `flag` 的值,若不为零,则接收消息;若为零,则知晓通信结束,跳出 `while(1)` 循环。父进程结束发送时,向子进程发送信号,然后等待子进程终止。子进程接收到信号后执行信号处理函数,将 `flag` 置零,便可跳出 `while(1)` 循环。

最后要感谢薛质老师的详细讲解,让我对进程和进程通信有了更为深入的理解。

A utils.h 文件

我将实验中用到的库包含、宏定义和工具函数封装成了 `utils.h` 头文件,内容如下。

```

1 #ifndef UTILS_H
2 #define UTILS_H
3
4 #include <errno.h>
5 #include <signal.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <string.h>
9 #include <sys/ipc.h>
10 #include <sys/msg.h>
11 #include <sys/sem.h>
12 #include <sys/shm.h>
13 #include <sys/types.h>
14 #include <sys/wait.h>

```

```
15 #include <time.h>
16 #include <unistd.h>
17
18 #define MSGKEY 1234
19 #define MSGTYPE 1
20 #define BUFFER_SIZE 1024
21
22 struct msgtype {
23     long mtype;
24     int mpid;
25     size_t msize;
26     char mtext[BUFFER_SIZE];
27 };
28
29 #define SHM_SIZE 1024
30 #define SHMKEY rand()
31 #define SEMKEY1 rand() + 1
32 #define SEMKEY2 rand() + 2
33
34 int creatsem(key_t key) {
35     int semid;
36     union semun {
37         int val;
38         struct semid_ds *buf;
39         unsigned short *array;
40     } semopts;
41     semid = semget(key, 1, IPC_CREAT | 0666);
42     if (semid == -1) {
43         perror("semget");
44     }
45     semopts.val = 1;
46     if (semctl(semid, 0, SETVAL, semopts) == -1) {
47         perror("semctl");
48     }
49     return semid;
50 }
51
52 static void semcall(int semid, int op) {
53     struct sembuf semopbuf;
54     semopbuf.sem_num = 0;
55     semopbuf.sem_op = op;
56     semopbuf.sem_flg = 0;
57     if (semop(semid, &semopbuf, 1) == -1) {
58         perror("semop");
59     }
60 }
61
62 void P(int semid) { semcall(semid, -1); }
63 void V(int semid) { semcall(semid, 1); }
64
65 #define ANSI_COLOR_RED "\x1b[31m"
```

```
66 #define ANSI_COLOR_GREEN "\x1b[32m"
67 #define ANSI_COLOR_YELLOW "\x1b[33m"
68 #define ANSI_COLOR_BLUE "\x1b[34m"
69 #define ANSI_COLOR_RESET "\x1b[0m"
70
71 #endif // UTILS_H
```

B Makefile 文件

本实验使用的工程文件 Makefile 如下。

```
1 CC = gcc
2 CFLAG = -Wall -O3
3
4 procfork: procfork.c utils.h
5     $(CC) $(CFLAG) -o procfork procfork.c
6
7 client: client.c utils.h
8     $(CC) $(CFLAG) -o client client.c
9
10 server: server.c utils.h
11     $(CC) $(CFLAG) -o server server.c
12
13 procshm: procshm.c utils.h
14     $(CC) $(CFLAG) -o procshm procshm.c
15
16 clean:
17     rm procfork client server procshm
```