hw3

消息队列实现

producer

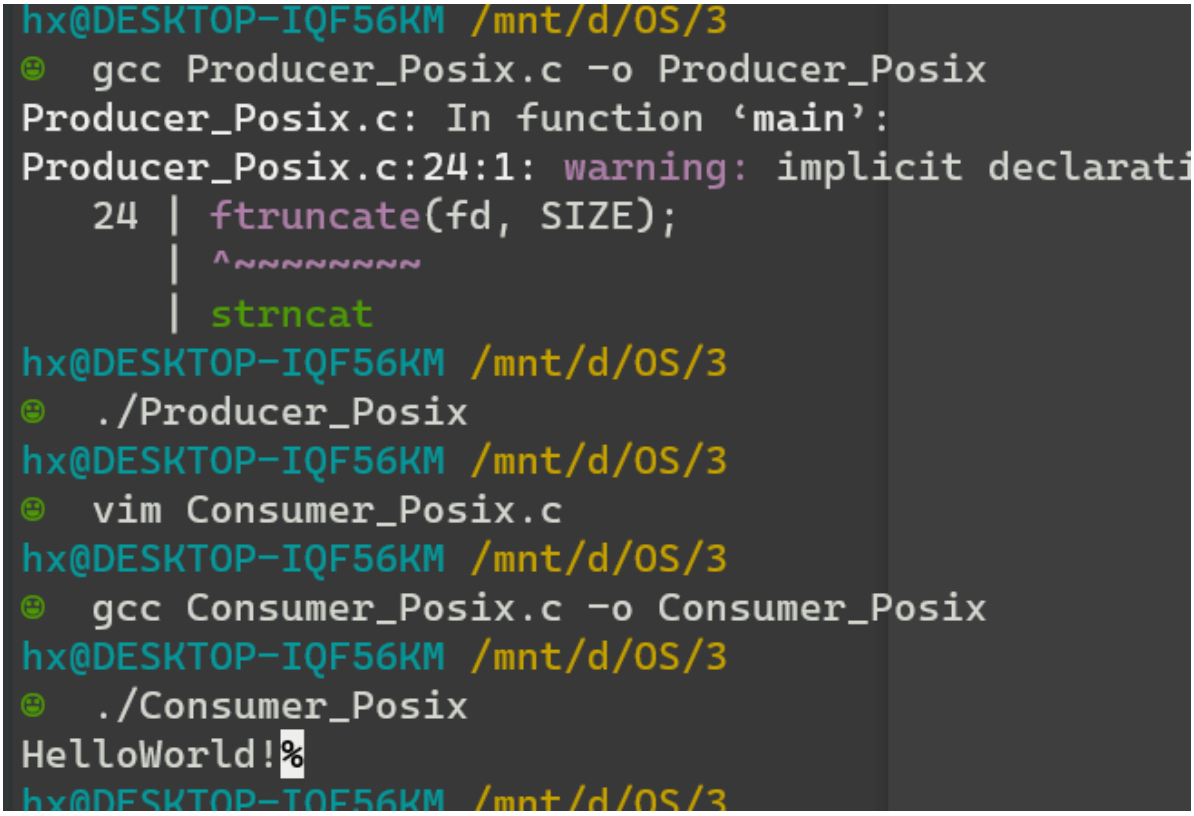
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
int main()
{
/* the size (in bytes) of shared memory object */
const int SIZE = 4096;
/* name of the shared memory object */
const char *name = "OS";
/* strings written to shared memory */
const char *message_0 = "Hello";
const char *message_1 = "World!";
/* shared memory file descriptor */
int fd;
/* pointer to shared memory obect */
char *ptr;
/* create the shared memory object */
fd = shm_open(name,O_CREAT | O_RDWR,0666);
/* configure the size of the shared memory object */
ftruncate(fd, SIZE);
/* memory map the shared memory object */
ptr = (char *)
mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
/* write to the shared memory object */
sprintf(ptr,"%s",message_0);
ptr += strlen(message_0);
sprintf(ptr,"%s",message_1);
ptr += strlen(message_1);
return 0;
}
```

consumer

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
int main()
{
    /* the size (in bytes) of shared memory object */
```

```c
    const int SIZE = 4096;
    /* name of the shared memory object */
    const char *name = "OS";
    /* shared memory file descriptor */
    int shm_fd;
    /* pointer to shared memory obect */
    void *ptr;
    /* open the shared memory object */
    shm_fd = shm_open(name, O_RDONLY, 0666);
    /* memory map the shared memory object */
    ptr = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
    /* read from the shared memory object */
    printf("%s", (char *)ptr);
    /* remove the shared memory object */
    shm_unlink(name);
    return 0;
}
```

结果



3.1 3.5

3.1 Using the program shown in Figure 3.30, explain what the output will be at LINE A.

依旧是5

3.5 When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process?

共享内存区段

3.8, 3.9, 3.12, 3.13, 3.14, 3.17

3.8 Describe the differences among short-term, medium-term, and long-term scheduling.

短期（CPU调度器）：从内存中准备执行的进程中选择一个进程，并将CPU分配给它。

中期（内存管理器）：从准备好的或阻塞的队列中选择一个进程，把它们从内存中换到磁盘上，然后再把它们换进去继续运行。

长期（作业调度器）：决定哪些作业从磁盘上的回存带入内存进行处理。

一个主要的区别在于其执行的频率。

3.9

作为对时钟中断的响应，操作系统保存当前执行进程的PC和用户堆栈指针，并将控制权转移到内核时钟中断处理程序。

时钟中断处理程序将其余的寄存器以及其他机器状态，如浮点寄存器的状态，保存在进程PCB中。

操作系统调用调度器来决定下一个要执行的进程。

然后操作系统从其PCB中检索下一个进程的状态，并恢复寄存器。这个恢复操作使处理器回到了这个进程之前被中断的状态，以用户模式权限执行用户代码。

3.12

16

3.13

`execlp()`函数属于`exec()`函数族

如果`execlp`调用成功,进程自己的执行代码就会变成加载程序的代码,`execlp()`后边的代码也就不会执行了

所以无法到达`printf("LINE J")`代码

3.14

A：0

B：2603

C：2603

D：2600

3.17

1, 2, 3, 4
因为子程序是父程序的副本，子程序的任何改变都会发生在它的数据副本中，不会反映在父程序中。因此，子代在X行输出的值是0, -1, -4, -9, -16。父代在Y行输出的值是0, 1, 2, 3, 4