

システムソフトウェア 課題 1

21B30362 佐久川泰輔

2023 年 11 月 10 日

1 問 1

1.1 コードの説明

以下は、作成した pingpong.c のメイン関数の中身である。

Listing 1 pingpong.c

```
1 int main(int argc, char *argv[]) {
2     if (argc != 2) {
3         fprintf(1, "usage: %s N\n", argv[0]);
4         exit(1);
5     }
6
7     // # of rounds
8     int n = atoi(argv[1]);
9
10    // tick value before starting rounds
11    int start_tick = uptime();
12
13    int pp[2], qq[2];
14    unsigned char buf[1];
15    unsigned int num = 0;
16
17    //create pipe
18    pipe(pp); //child -> parent
19    pipe(qq); //parent -> child
20
21    int status;
22    if(fork() > 0){
23        //parent process
24        close(qq[0]);
25        close(pp[1]);
26
27        for(int i=0;i<n;i++){
28            buf[0]=(unsigned char)num%256;
29            write(qq[1], buf, 1);
```

```

30         read(pp[0],buf,1);
31         num = (int)buf[0];
32         num++;
33     }
34
35     //wait child process
36     wait(&status);
37
38     // tick value after ending rounds
39     int end_tick = uptime();
40     // print # of ticks in N rounds
41     printf("# of ticks in %d rounds: %d\n", n, end_tick - start_tick);
42     exit(0);
43 }else{
44     //child process
45     close(pp[0]);
46     close(qq[1]);
47
48     for(int i=0;i<n;i++){
49         read(qq[0],buf,1);
50         num = (int)buf[0];
51         num++;
52         buf[0]=(unsigned char)num%256;
53         write(pp[1],buf,1);
54     }
55 }
56 exit(0);
57 }

```

まず、11 行目でパイプを作成する前に開始時のティック数を記録する。

次に、親プロセスと子プロセスの間に通信を行うためのパイプを作成する。ただし、パイプ pp は子プロセスから親プロセスの、パイプ qq はその逆を行うために使用する。

その後、22 行目で fork() を実行し、子プロセスを生成する。

23 行目から 42 行目は親プロセスの処理を行うコードである。27 行目から 33 行目で、n 回読み書きを行う。この際、書いた後に読むようにする。

44 行目から 54 行目は子プロセスの処理を行うコードである。48 行目から 54 行目で、親プロセスと同様に読み書きを行う。この際、親プロセスとは逆に読んだ後に書くようにする。

親プロセスでは最後に、子プロセスの終了を待った後に終了時のティック数を記録する。

また、データの送信には要素数 1 の unsigned char 型の配列を使用しているが、別プロセスからのデータを読み込んだ際は、これを unsigned int 型にキャストし (28、52 行目)、書き込む際にもキャストしている (31、50 行目)。この際、unsigned int から unsigned char にキャストする時には、データがオーバーフローしないように 256 で割った後にキャストしている。

1.2 実行結果

実行結果は以下である。



```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
[$ pingpong 100
# of ticks in 100 rounds: 1
[$ pingpong 1000
# of ticks in 1000 rounds: 4
[$ pingpong 10000
# of ticks in 10000 rounds: 47
$
```

2 問 2


2.1 コードの説明

まず、第 10 回の講義で説明していただいたシステムコール `gettimeofday()` を使用できるように、ブランチ `rtc` を `hw1` にマージした。

次に、問 1 で使用したコードのうち、`uptime()` を使用している箇所 (1-1 中のコードの 11,39 行目) を `gettimeofday()` に書き換えた。

2.2 実行結果

実行結果は以下である。



```
xv6-riscv — xv6@20d2e33ef090: ~/xv6-riscv — com.docker.cli • docker run -it --rm -v ~/WebGL/xv6-riscv/home/xv6-riscv wtakuo/xv6-env — 163x10

hart 2 starting
hart 1 starting
init: starting sh
[$ pingpong 100
# of timesec in 100 rounds: 0
[$ pingpong 1000
# of timesec in 1000 rounds: 0
[$ pingpong 10000
# of timesec in 10000 rounds: 5
$
```

3 参考文献

プロセス間通信について、以下の Web サイトを参考にした。

親プロセス・子プロセス間通信. <https://qiita.com/penkopenko/items/f28d0b5ef404afd339fa>.
2023-11-5