

## Network System Programming Quiz 2

### 2025/10/28 Tuesday

1. The core task is to implement a one-to-many concurrency pattern using processes: a single Producer and up to  $k = 50$  Consumers. The Producer must write sequentially generated random integers (1-9) into a shared file every  $t$  seconds (where  $t$  can be as short as 0.1s). Communication must use a signal scheme to notify consumers when new data is available, and the Producer must immediately overwrite the previous data. Both  $t$  and  $k$  must be accepted via command-line arguments. Each Consumer must maintain a cumulative sum of all received integers. Transmission ends when the Producer writes a special termination character ('#'); upon receipt, each Consumer must print its ID (1 to  $k$ ) and its final cumulative sum to standard output. Linux system calls like *fork()*, *pipe()*, *getpid()*, *pause()*, *setitimer()*, *signal()*, *kill()*, and process grouping *setpgid()*, *getpgid()*, *killpg()* are encouraged for implementation.

Note: Leveraging process groups with calls like *setpgid()* and *killpg()* may optimize signal distribution.

```
// The following code snippets are provided for your reference.
#include <unistd.h>
// Set the calling process as the leader of a new process group
setpgid(0, 0);
// Move a child process (with PID child_pid) to a new process group (with PGID pid)
setpgid(child_pid, pid);
// Generate a random number between 1 and 9
srand(time(NULL)); // Seed the random number generator with current time
int random_number = (rand() % 9) + 1;
```

Usage: **pro\_con** *num\_consumer interval filename*

(If the *filename* is omitted from the input, it must be treated as a hardcoded or pre-established constant known universally by the Producer and every Consumer.)

For example: (What will happen with large  $k$ , small  $t$ ?)

```
$ pro_con 10 3 datafile
```

```
Consumer 9: 15
```

```
Consumer 1: 15
```

```
Consumer 4: 15
```

```
Consumer 7: 15
```

```
Consumer 3: 15
```

```
.....
```

Key insights:

- The Producer forks a leader Consumer, which, in turn, forks to generate the rest of the Consumers. The leader assigns the Consumer IDs.
- Process Grouping: What is the mechanism for setting or notifying the Process Group ID?
- What method should I use to inform users about the shared filename?