CS39002 Operating Systems Laboratory
Spring 2025

Lab Assignment: 2
Date of submission: 15–Jan –2025

Inter-process communication using signals

Think of n children $C_1, C_2, C_3, \ldots, C_n$ standing in a circle, and playing a game with their parent P standing at the center of the circle. P throws a ball to the children in the circular order. If the child (say, C_i) to which the ball is thrown can catch the ball, then C_i continues to play. If C_i misses the ball, then C_i goes out of the game. After each throw, the ball comes back to P who then throws the ball to the next (in the circular order) child who is not yet out of the game. Eventually, $n - 1$ children miss and go out of the game. The remaining child wins the game.

You need to implement this game as a multi-process application where the processes communicate with one another by signals. Write two programs *parent.c* and *child.c* to simulate the working of the parent process and of each child process. Suppose that these two programs are compiled to the executable files *parent* and *child*. The program *parent* (which simulates P) is run with one command-line argument n (the number of child processes). P creates n child processes C_i which exec *child* i for $i = 1, 2, 3, \ldots, n$. P also writes, in a text file *childpid.txt*, the child count n and the PIDs of the n child processes created by P. Each child first waits for some time (like one second) for P to finish writing to *childpid.txt*. After this wait, each child reads n and the n PIDs from the text file *childpid.txt*. After the child creation, P waits for some time (like two seconds) so that each child process gets time to read *childpid.txt*. These waits may be implemented by *sleep*() or *usleep*(), but after this, no waits based on these functions will be allowed.

The parent P then enters a loop which continues until only one child is left as the player. Each child C_i, on the other hand, enters an infinite loop. The body of each loop should contain the single system call `pause()` which lets the calling process wait until it receives a signal (this way, a busy wait is avoided). The game of throwing balls and catching/missing throws will be implemented by sending signals. In this assignment, we use the three signals `SIGUSR1`, `SIGUSR2`, and `SIGINT` only. P starts the game by sending `SIGUSR2` to C_1.

For a child process C_i, receiving `SIGUSR2` implies that a throw is made to it. It then randomly decides whether it catches the ball (with probability 0.8) or misses the ball (with probability 0.2). If C_i catches the ball, it sends `SIGUSR1` to P. If C_i fails to catch the ball, it sends `SIGUSR2` to P. Depending on the type of the signal received from the child C_i (to which the throw is made), the parent knows whether that child continues to play the game or is out of the game. P records this information. The child too records its own status.

After the outcome of a throw is recorded as explained above, P initiates a printing of the current status of all the n players. Since P has all the necessary information, it can do that printing itself. However, as a mandatory part of this assignment, this printing should be done collaboratively by the child processes. This is achieved by sending `SIGUSR1` to the child processes in turn. Recall that to a child process, receiving `SIGUSR2` means that a throw is made to it. On the other hand, the reception of `SIGUSR1` instructs that child to print its current status. The possible status of a child are PLAYING (written as `....`), CATCHMADE (written as `CATCH`), CATCHMISSED (written as `MISS`), and OUTOFGAME (written as blank). See the sample at the end to know the format of printing. Strictly follow this format.

P initiates the printing by sending `SIGUSR1` to C_1. For each $i < n$, C_i prints its status, and then sends `SIGUSR1` to the next child C_{i+1}. The last child C_n prints its status, but does not send `SIGUSR1` to any other process. However, C_n takes part in the synchronization activity in a different manner. Until all the child processes finish writing their status, the parent P must wait before it can make the throw to the next playing child. Currently, you know only a few synchronization primitives, so let this wait be accomplished by *waitpid*(). Before sending `SIGUSR1` to C_1, P forks a dummy child process D, and writes the PID of D in a text file *dummycpid.txt*. After sending `SIGUSR1` to C_1, P waits until D exits. When C_n is done printing its status, it reads the PID of D from the file *dummycpid.txt*, and sends `SIGINT` to D. Write *dummy.c* (the code for *D*) that enters an infinite loop of `pause()` at the beginning of its *main*(). It is not meant to do any useful work except putting an end to the wait of the parent *P*.

When *D* exits, *P* wakes up, and works out the next playing child process C_{next} to which the throw is to be made. Recall that *P* maintains the information of the status of all child processes. *P* should also keep track of the child process to which the last throw is made. So *P* can determine C_{next} easily. *P* then sends `SIGUSR2` to C_{next}, and the game continues as explained above.

After $n - 1$ child processes miss throws, the parent sends `SIGINT` one by one to all of the *n* child processes. Only the last playing child process prints a happy message, and exits (see the format in the sample). The other processes exit without printing anything.

The sequencing of the throw-and-print cycle must be implemented only by signals (and by *waitpid*() in one situation). No other synchronization mechanism is allowed. Use *fflush*(*stdout*); to avoid garbled output. But do not use any *sleep* or *usleep* calls (except only at the very beginning, that is, before the game starts).

You may use the following *makefile*.

```
all:
	gcc -Wall -o parent parent.c
	gcc -Wall -o child child.c
	gcc -Wall -o dummy dummy.c

run: all
	./parent 10

clean:
	-rm -f parent child dummy childpid.txt dummycpid.txt
```

Submit a zip/tar/tgz archive containing the files *parent.c*, *child.c*, *dummy.c*, and *makefile*.

Sample Output

```
$ make run
gcc -Wall -o parent parent.c
gcc -Wall -o child child.c
gcc -Wall -o dummy dummy.c
./parent 10
Parent: 10 child processes created
Parent: Waiting for child processes to read child database

       1       2       3       4       5       6       7       8       9      10
+----------------------------------------------------------------------------------------+
|    CATCH   ....    ....    ....    ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    CATCH   ....    ....    ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    CATCH   ....    ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    CATCH   ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    CATCH   ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....    CATCH   ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....    ....    CATCH   ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....    ....    ....    CATCH   ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....    ....    ....    ....    CATCH   ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....    ....    ....    ....    ....    CATCH       |
+----------------------------------------------------------------------------------------+
|    CATCH   ....    ....    ....    ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    CATCH   ....    ....    ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    CATCH   ....    ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    CATCH   ....    ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    CATCH   ....    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....    MISS    ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....            CATCH   ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....            ....    CATCH   ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....            ....    ....    CATCH   ....        |
+----------------------------------------------------------------------------------------+
|    ....    ....    ....    ....    ....            ....    ....    ....    CATCH       |
+----------------------------------------------------------------------------------------+
|    CATCH   ....    ....    ....    ....            ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....    MISS    ....    ....    ....            ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....            CATCH   ....    ....            ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....            ....    CATCH   ....            ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....            ....    ....    MISS            ....    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....            ....    ....                    MISS    ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....            ....    ....                            CATCH   ....    ....        |
+----------------------------------------------------------------------------------------+
|    ....            ....    ....                            ....    CATCH   ....        |
+----------------------------------------------------------------------------------------+
|    ....            ....    ....                            ....    ....    CATCH       |
+----------------------------------------------------------------------------------------+
|    MISS            ....    ....                            ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|                    CATCH   ....                            ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|                    ....    MISS                            ....    ....    ....        |
+----------------------------------------------------------------------------------------+
|                    ....                                    CATCH   ....    ....        |
+----------------------------------------------------------------------------------------+
|                    ....                                    ....    CATCH   ....        |
+----------------------------------------------------------------------------------------+
|                    ....                                    ....    ....    MISS        |
+----------------------------------------------------------------------------------------+
|                    CATCH                                   ....    ....                |
+----------------------------------------------------------------------------------------+
|                    ....                                    CATCH   ....                |
+----------------------------------------------------------------------------------------+
|                    ....                                    ....    CATCH               |
+----------------------------------------------------------------------------------------+
|                    CATCH                                   ....    ....                |
+----------------------------------------------------------------------------------------+
|                    ....                                    CATCH   ....                |
+----------------------------------------------------------------------------------------+
|                    ....                                    ....    CATCH               |
+----------------------------------------------------------------------------------------+
|                    CATCH                                   ....    ....                |
+----------------------------------------------------------------------------------------+
|                    ....                                    MISS    ....                |
+----------------------------------------------------------------------------------------+
|                    ....                                            MISS                |
+----------------------------------------------------------------------------------------+
       1       2       3       4       5       6       7       8       9      10
+++ Child 3: Yay! I am the winner!
$
```