CSS 430 Operating Systems
Program 1 Report

Part 1: Linux System Programming

1. **Processes.cpp**

   processes.cpp is a program that receives an argument from the user and returns the number of currently running processes of that argument name at the time of execution.

```
$ g++ processes.cpp -o processes
$ ./processes <input>
```

2. **Algorithm**

   The program completes the task by consecutively creating child processes from the parent and executing one command at a time.

   A parent process forks a child process and waits for it to return. The child forks a grand child process and creates a pipe (pipeFD1) to communicate with it. The grand child forks a great grand child process and create a pipe (pipeFD2) to communicate with it.

   The great grandchild redirects the standard output to be written onto pipeFD2 and executes "`ps -A`". When the great grand child exits, the grand child redirects the standard input to pipeFD2 and standard output to pipeFD1. Then, grand child executes "`grep <argv[1]>`". After the grand child exits, the child redirects standard input to pipeFD1 and executes "`wc -l`". After displaying output to the console, the child exits. The interactions between the processes are shown in Figure 1.

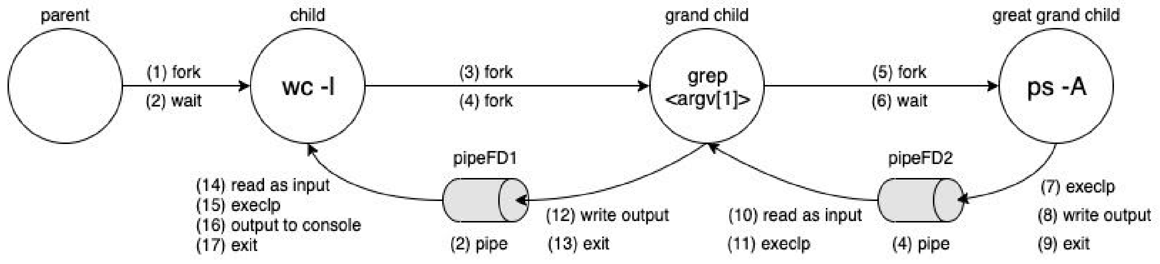
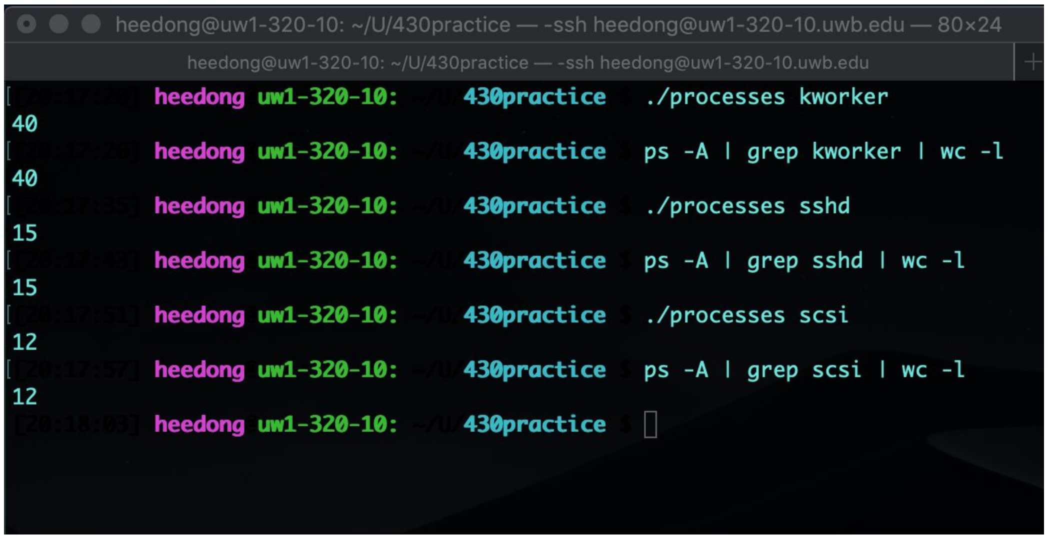


Figure 1. Process tree for processes.cpp

3. **Tests**

   To test this program, we can compare the output of the two commands:

```
$ ./processes <input>
$ ps -A | grep <intput> | wc -l
```



```
heedong@uw1-320-10: ~/U/430practice — -ssh heedong@uw1-320-10.uwb.edu — 80×24
heedong@uw1-320-10: ~/U/430practice — -ssh heedong@uw1-320-10.uwb.edu
[20:17:20] heedong uw1-320-10: ~/U/430practice $ ./processes kworker
40
[20:17:26] heedong uw1-320-10: ~/U/430practice $ ps -A | grep kworker | wc -l
40
[20:17:35] heedong uw1-320-10: ~/U/430practice $ ./processes sshd
15
[20:17:43] heedong uw1-320-10: ~/U/430practice $ ps -A | grep sshd | wc -l
15
[20:17:51] heedong uw1-320-10: ~/U/430practice $ ./processes scsi
12
[20:17:57] heedong uw1-320-10: ~/U/430practice $ ps -A | grep scsi | wc -l
12
[20:18:03] heedong uw1-320-10: ~/U/430practice $
```

Figure 2. Output comparison for processes.java

Part 2: ThreadOS Shell Design

1. **Shell.java**

   Shell.java is a program that behaves similarly to an actual shell. It reads in series of keyboard inputs from the user and executes them if they are valid. User can use this program to execute programs concurrently by delimiting commands with "&", or sequentially by delimiting them with ";". If a command is not delimited by either of those, the program assumes that it may be run sequentially.

   This program does not support pipe ("|") or redirection("<", ">").

2. **Algorithm**

   This program takes in commands of user input that are delimited by “&” and execute them concurrently, and that are delimited by “;” and execute them sequentially. When the program is executing a series of commands that are delimited by “;”, it waits until all of the processes are done executing to execute the next set of commands. The program flow is shown in Figure 3.

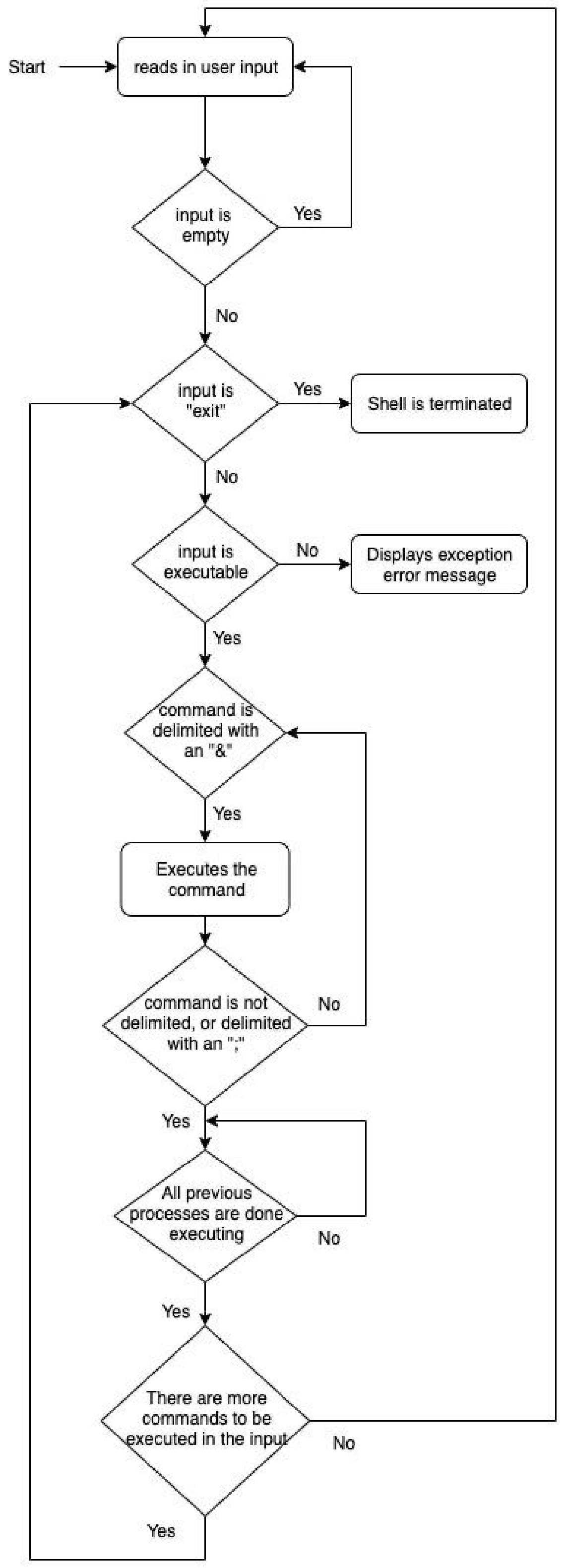

Start
reads in user input
input is empty
Yes
No
input is "exit"
Yes
Shell is terminated
No
input is executable
No
Displays exception error message
Yes
command is delimited with an "&"
Yes
Executes the command
command is not delimited, or delimited with an ";"
No
Yes
All previous processes are done executing
No
Yes
There are more commands to be executed in the input
No
Yes

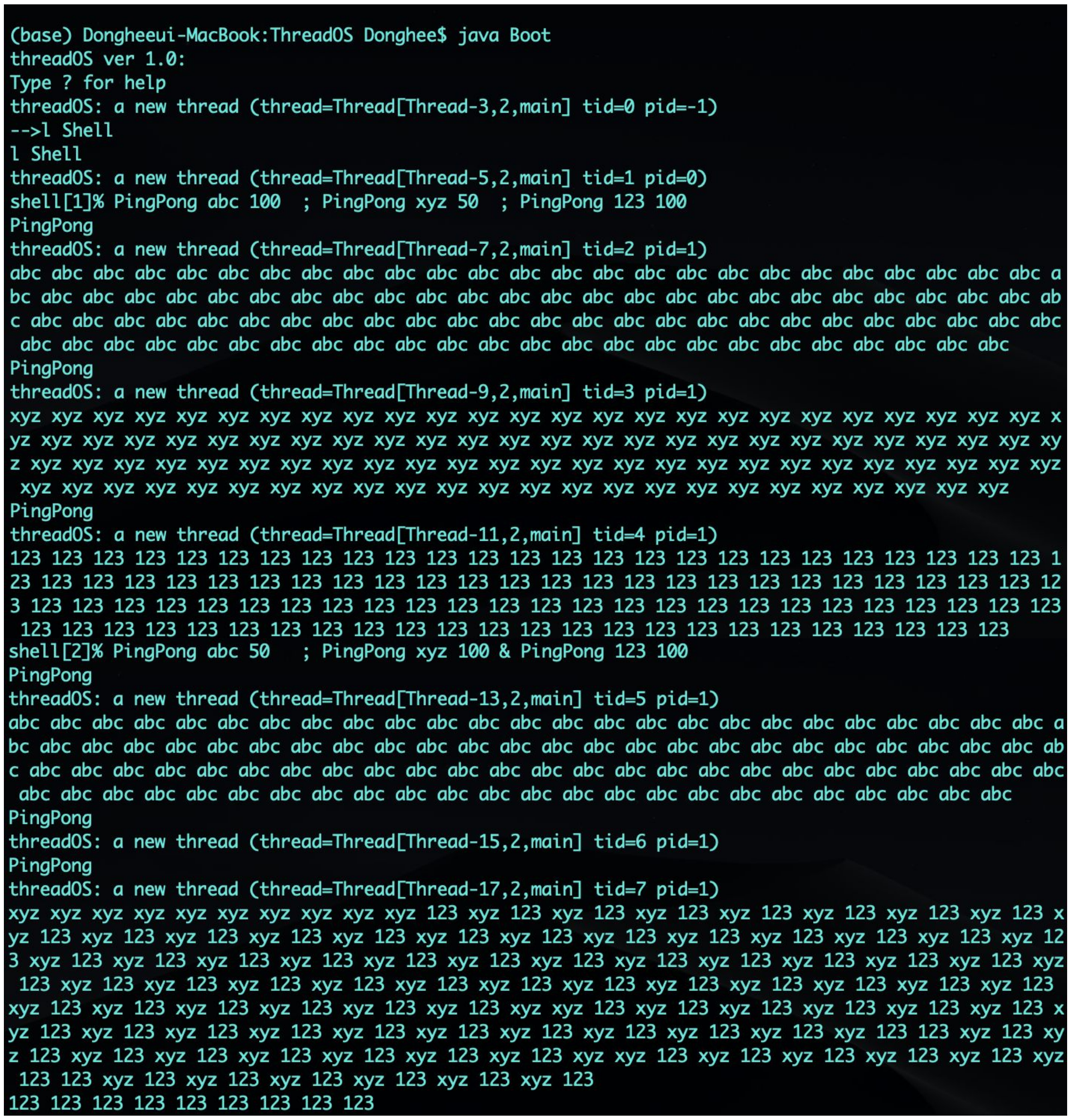
Figure 3. Flowchart for Shell program

3. Tests

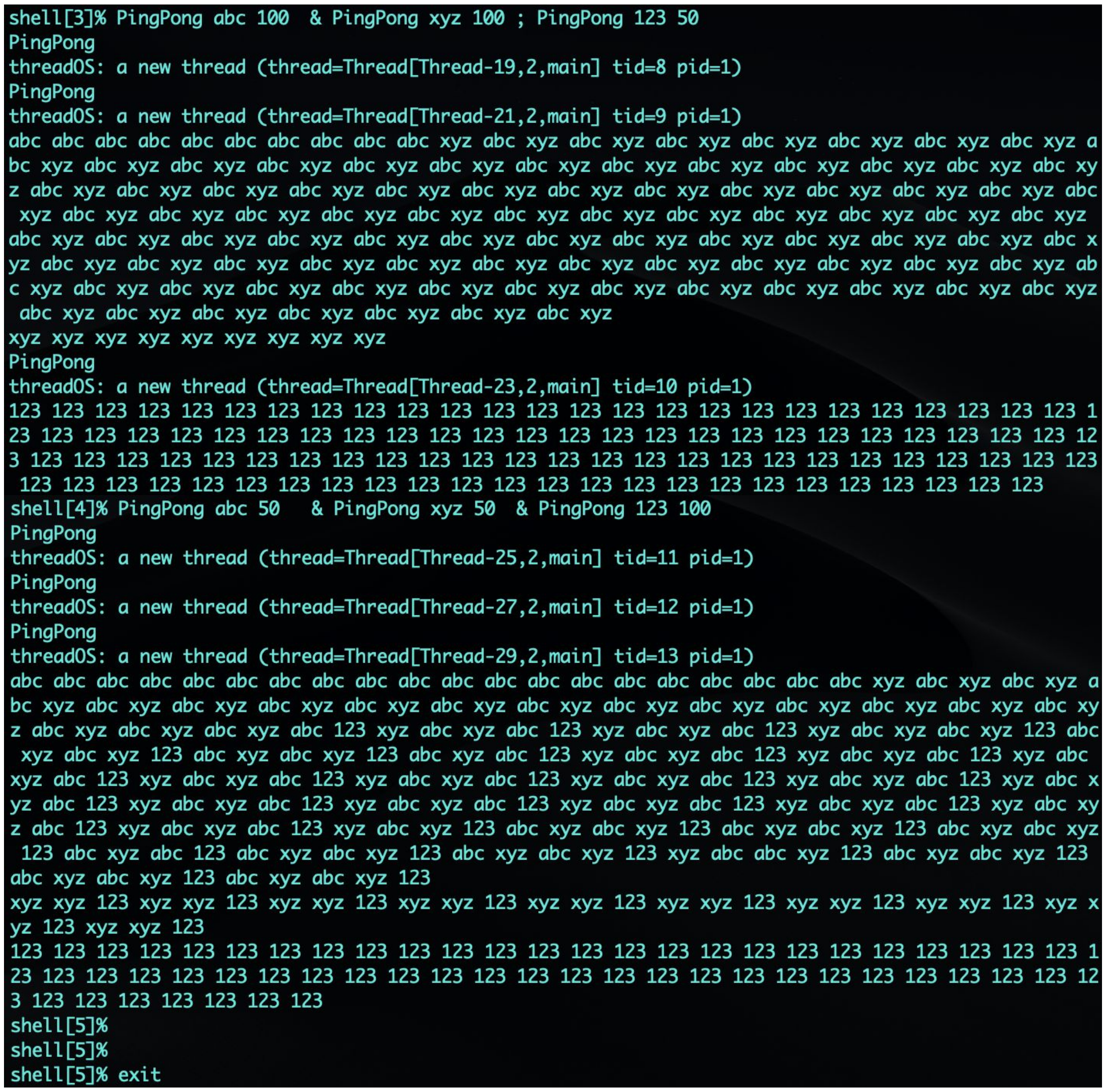
Tests can be done by executing PingPong.java on Shell that takes in two arguments: a string and an int. Users can run several instances of PingPong with different strings using “&” and “;” to test if the programs are running sequentially or concurrently. Th

```
Shell[1]% PingPong abc 100  ; PingPong xyz 50  ; PingPong 123 100
Shell[2]% PingPong abc 50   ; PingPong xyz 100 & PingPong 123 100
Shell[3]% PingPong abc 100  & PingPong xyz 100 ; PingPong 123 50
Shell[4]% PingPong abc 50   & PingPong xyz 50  & PingPong 123 100
```

The output of these test cases is shown in Figure 3.



```
(base) Dongheeui-MacBook:ThreadOS Donghee$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
shell[1]% PingPong abc 100  ; PingPong xyz 50  ; PingPong 123 100
PingPong
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc a
bc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc ab
c abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
 abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
PingPong
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz x
yz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xy
z xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
 xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
PingPong
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 1
23 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 12
3 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
shell[2]% PingPong abc 50   ; PingPong xyz 100 & PingPong 123 100
PingPong
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc a
bc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc ab
c abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
 abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
PingPong
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 x
yz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 12
3 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz
 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123
xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 x
yz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 123 xyz 123 xy
z 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz
 123 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123
123 123 123 123 123 123 123 123 123
```



```
shell[3]% PingPong abc 100  & PingPong xyz 100 ; PingPong 123 50
PingPong
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
abc abc abc abc abc abc abc abc abc abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz a
bc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xy
z abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc
 xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz
abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc x
yz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz ab
c xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz
 abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz
PingPong
threadOS: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=1)
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 1
23 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 12
3 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
shell[4]% PingPong abc 50   & PingPong xyz 50  & PingPong 123 100
PingPong
threadOS: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-29,2,main] tid=13 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc xyz abc xyz abc xyz a
bc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xy
z abc xyz abc xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc xyz 123 abc
 xyz abc xyz 123 abc xyz abc xyz 123 abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc
xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc x
yz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xy
z abc 123 xyz abc xyz abc 123 xyz abc xyz 123 abc xyz abc xyz 123 abc xyz abc xyz 123 abc xyz abc xyz
 123 abc xyz abc 123 abc xyz abc xyz 123 abc xyz abc xyz 123 xyz abc abc xyz 123 abc xyz abc xyz 123
abc xyz abc xyz 123 abc xyz abc xyz 123
xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz x
yz 123 xyz xyz 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 1
23 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 12
3 123 123 123 123 123 123 123
shell[5]%
shell[5]%
shell[5]% exit
```

Figure 4. Test output from Shell program