Documentation for processes.cpp

Processes is a program that simulates a unix shell with a fixed set of commands. It takes a single argument from the command line that is used as part of an otherwise predefined, piped set of commands. Specifically, the commands “ps -A | grep argv[1] | wc -l” are simulated, where argv[1] contains a user input argument.

The program begins by checking to see the user has provided the required input. Each command (ps -A, grep argv[1], and wc -l) is executed in a separate child, grandchild, and great grandchild process, in reverse order. The three processes are connected by two pipes. The parent process forks a child process and waits for it to finish. That child opens a pipe and forks a grandchild process and waits for it to finish. That grandchild opens a pope and forks a great grandchild process and waits for it to finish. The great grandchild then executes the first command, ps -A, and redirects its output to the first pipe. The grandchild then reads from the first pipe and executes the second command, grep argv[1], and redirects its output to the second pipe. The child then reads from the second pipe and executes the last command, wc -l, and dumps its output to the console. Finally the parent process where the whole show began regains control and the program ends.

Output for processes.cpp

```
retlaw@uw1-320-13:~/430/Program1$ ./processes kworker
35
retlaw@uw1-320-13:~/430/Program1$ ps -A | grep kworker | wc -l
35
retlaw@uw1-320-13:~/430/Program1$ ./processes sshd
4
retlaw@uw1-320-13:~/430/Program1$ ps -A | grep sshd | wc -l
4
retlaw@uw1-320-13:~/430/Program1$ ./processes scsi
12
retlaw@uw1-320-13:~/430/Program1$ ps -A | grep scsi | wc -l
12
```

Documentation for Shell.java

Shell begins by initializing a count variable to 1 that will be used in displaying the number of commands executed. A default empty constructor and an overloaded constructor taking a String array are provided to prevent the program from breaking regardless of whether the user enters zero or several arguments, although nothing is done by either constructor to initialize the program.

Since Shell extends Thread, its main functionality is implemented in the run() method. The run() method consists of a command loop that loops forever accepting commands from the user until they type “exit”. Commands are accepted via a helper method readCmdLine() that displays a command prompt and reads input from the keyboard. Each time a user enters a command, the command count is incremented, and a new prompt is displayed with the updated command number. If a user fails to enter a command, then the prompt is displayed again with the same command number.

After a user enters a line of commands, the command loop calls a second helper method processCmdLine() to process it. This method iterates through each user-input command set and separates the command-argument sets from the punctuating ‘;’ and ‘&’ characters. Until one of the punctuating characters is reached, the method accumulates each command-argument into a String. It accomplishes this by looking ahead one element in the command array to see if the next element is a punctuation character. Obviously, the method cannot run to the end of the array while looking one ahead because it will look out of bounds on the last element. So, after it stops one index short of the end of the array, the method performs a last check to see whether or not the last value is a command, and if it is, then it executes it concurrently. The method knows the last element is a concurrent command if it isn’t a punctuation mark because if it were a ‘&’ then it would have been picked up already when the loop was in its last look-ahead iteration.

How to test Shell.java

Copy and compile Shell.java into the directory where ThreadOS lives. To start ThreadOS, type “java Boot” and hit enter. When ThreadOS is ready, type “l Shell” and hit enter to load the Shell program into ThreadOS. To execute commands in Shell, type one or more command name sets, each containing a command name and zero or more arguments, with each set separated by a ‘;’ or a ‘&’. If the command set is punctuated with a ‘;’, then it will complete its execution before anything after it is allowed to run. If a command set is punctuated by a ‘&’, then it will run concurrently while the next command set is allowed to run, or if no command set is present, while the Shell command prompt returns. Finally, if the last command set is not punctuated, it is assumed that it is punctuated with a ‘;’, and the command set blocks until its execution is finished.

Output for Shell.java

```
retlaw@uw1-320-13:~/430/ThreadOS$ 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[0]% PingPong abc 100  ; PingPong xyz 50  ; PingPong 123 100
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 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 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
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 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 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
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
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123

Shell[1]% PingPong abc 50   ; PingPong xyz 100 & PingPong 123 100
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 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 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
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
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 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 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 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
123 123 123 123 123 123 123 123 123

Shell[2]% PingPong abc 100  & PingPong xyz 100 ; PingPong 123 50
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
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 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 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 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
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
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123

Shell[3]% PingPong abc 50   & PingPong xyz 50  & PingPong 123 100
threadOS: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=1)
threadOS: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=1)
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 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 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 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
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 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
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 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
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123

Shell[4]% exit
goodbye
Superblock synchronized
-->
```