**OPERATING SYSTEM (PRACTICALS) – FALL 2012**

**EXPERIMENT 3 – PROCESS**

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE: 5th Oct 2012** | | **Students Names: Capt Sumeer Riaz** | |
| **Marks Obtained: \_\_\_\_\_** | | **COURSE: BESE 16 B** | |
| **Deadline: 1400 hrs 8th Oct 2012** | | **Instructor: Engr. Umar Mahmud** | |
| 1. | **Objectives:**   1. Creating process in Java. 2. Destroying processes in Java | |  |
| 2. | **Time Required:** 3 hrs | |  |
| 3. | **Software Required:**   1. Windows OS 2. NetBeans 7.2/JCreator | |  |
| 4. | **Process** is a program in execution; process execution must progress in sequential fashion. A process includes a program counter , stack and data section | |  |
| 5. | Check the PID of WINWORD.EXE on your machine and write it here  **Answer:** **1572** | |  |
| 6. | Close the WINWORD application, restart WINWORD after 1 minute and write the PID here  **Answer:** **1921.** | |  |
| 7. | What is your observation in points 5 and 6?  **Answer:** When same application was openedsimultaneously twice or thrice and at same time or after few seconds, Process IDs were different, even same application was opened. | | (2) |
| 8. | **Getting PID of JVM through Java:** For the given code show what is the output  import java.util.\*;  import java.io.\*;  import java.lang.management.\*;  public class JavaPID {  public static void main(String args[]){  try{  System.out.println(ManagementFactory.getRuntimeMXBean().getName());  }  catch (Exception t){  t.printStackTrace();  }  }  } | |  |
| 9. | Describe what you understand from the output in point 8.  **Answer:** PID **29490** and Ccomputer name : **WIN-IQKINNLQF33** was displayed. | | (2) |
| 10. | In command prompt type **mspaint** and what is the output?  **Answer:** Application paint will be opened. | | (1) |
| 11. | **Creating a Process in Java:** A number of methods exist in Java to create a process. We will follow the method through Java Process class. For the given code  import java.util.\*;  import java.io.\*;  import java.lang.management.\*;  public class JavaExec {  public static void main(String args[]){  try {  System.out.println(ManagementFactory.getRuntimeMXBean().getName());  Runtime rt = Runtime.getRuntime();  Process proc = rt.exec("mspaint"); // execute mspaint  Thread.sleep(5000); // system sleeps for 5 seconds  proc.destroy(); // close mspaint  }  catch (Exception t){  t.printStackTrace();  }  }  } | |  |
| 12. | What is the output of point 11?  **Answer:** Ms Paint was opened for exactly five seconds. And after the mentioned time, application was closed automatically. | | (2) |
| 13. | Now open Calculator and Freecell application using the same method | |  |
| 14. | To get the exit value modify the code as follows  import java.util.\*;  import java.io.\*;  import java.lang.management.\*;  public class JavaExec {  public static void main(String args[]){  try {  System.out.println(ManagementFactory.getRuntimeMXBean().getName());  Runtime rt = Runtime.getRuntime();  Process proc = rt.exec("mspaint"); // execute mspaint  **Process proc1 = rt.exec("calc");**  Thread.sleep(5000); // system sleeps for 5 seconds  proc.destroy(); // close mspaint  **proc1.destroy();**  int exitVal = proc.waitFor();  System.out.println("Process exitValue: " + exitVal);  }  catch (Exception t){  t.printStackTrace();  }  }  } | |  |
| 15. | What Is The Output Of Point 14?  Answer: **Ms**-**Paint** and **Calculator** applications were opened accordingly and were closed after exactly five seconds. And “Process exitVal: 0” was displayed in the output window. | |  |
| 16. | Re-execute point 14 but close the application in windows before the time expires. Write the output here.  **Answer:** “Process exitVal: 1” was displayed in the output | |  |
| 17. | What is your observation in Points 15 and 16?  **Observation:**  1. exitVal is ‘0’ when application was closed after the time expired (5 seconds).  2. exitVal is ‘1’ when application was closed before the time expired.  **Reason:** There are conventions for what sorts of status values certain programs should return. The most common convention is **simply ‘0’ for success** and **‘1’ for failure.** When a program exits, it returns to the parent process a small amount of information about the cause of termination, using the *exit status*. This is a value between 0 and 255 that the exiting process passes as an argument to exit. | | (2) |
| 18. | What did you learn in this experiment? | | (1) |

**Answer:** This lab has opened up mind with urge to find other shell commands to

execute with ‘**run window’**, ‘[**ms dos prompt**](http://www.google.com.pk/search?hl=en&biw=1280&bih=859&sout=0&tbm=isch&sa=X&ei=4JpxUIe5Do_SrQfOtoF4&spell=1&q=ms+dos+prompt&ved=0CD8QvwUoAA)**’** and as well as with the help

of java ‘**NetBeans 7.2/JCreator’**. This lab has provided interaction and handling

of NetBeans 7.2/JCreator in depth as far as shell commands are concerned. Opening

different applications with different commands as ‘mspaint’, ‘calc’, ‘appwiz.cpl’ and

other so many other commands proc.waitfor(); etc, is interesting.