**Header: Grade: FOR /20 DES /40 EXP /30 ORI /10 / TOT \_100**

*Lab # 1*

*CS 5390 Summer 2017, Date of Submission: 06/30/17*

*Students Names: Adeel Malik (amalik@utep.edu)*

*Instructor: Yadira Jacquez*

**Section 1: Effort: 10 hours**

- Planning and preparation: 2 hours

- Experiment: 6 hours (on simulator)

- Report writing: 2 hours

**Section 2: Objectives**

The objective of this experiment is to understand the concepts of Task operations. Understanding debugger, shell commands and resulting behavior. Recording and documenting the steps involved and the difference between steps after changing different parameters. After completing this lab, one should be able to perform commonly used operations related to tasks.

**Section 3: Procedures and Results**

**A1: Follow the procedure from previous lab to create a new project and add the source code below to the project.**

1. File, New VxWorks Downloadable Kernel Module Project. Project name “spawn\_tasks”, Next Next with default options and Finish.
2. Import the spawn\_tasks.c file, right click spawn\_tasks in the project explorer and import, select General-File System and Next, Locate file and Finish.
3. Right click spawn\_tasks in the Project Explorer and Build Project.

**A2: Execute printing from the shell. How do you do it, where are the results shown?**

1. Launch Host Shell, load spawn\_tasks.o file, execute the following command:

**-> printing (3)**

I am task 337901872, iter: #0

I am task 337901872, iter: #1

I am task 337901872, iter: #2

value = 0 = 0x0

Results shown in the Host Shell

**A3. Spawn the printing from the shell using sp, repeat, and period commands. Use WindSh help function to find out the syntax of these commands. Observe the output and the status of executing tasks. Note the naming convention for the spawned tasks. Where are the results shown?**

1. **Spawn the printing**

|  |  |
| --- | --- |
| **Shell** | **Target Console** |
| **-> sp (printing, 3)**  task spawned: id = 0x1423f930, name = s44u0  value = 337901872 = 0x1423f930 | -> I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2 |

1. **Repeat (Repeating the function 3 times)**

|  |  |
| --- | --- |
| **Shell** | **Target Console** |
| **-> repeat (3, printing, 3)**  task spawned: id = 0x1423f930, name = s44u1  value = 337901872 = 0x1423f930 | -> I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2  I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2  I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2 |

1. **Period (Running the function periodically)**

|  |  |
| --- | --- |
| **Shell** | **Target Console** |
| **-> period (5, printing, 3)**  task spawned: id = 0x1423f930, name = s44u2  value = 337901872 = 0x1423f930  To stop the task, use the following command, otherwise it will keep going after every 5 seconds.  **-> td(s44u2)**  value = 0 = 0x0 | -> I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2  I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2  I am task 337901872, iter: #0  I am task 337901872, iter: #1  I am task 337901872, iter: #2  ….  Ongoing after every 5 seconds. |

**A4. Execute spawn\_tasks(50). Observe currently executing tasks in various state using i shell function (you need to do it very fast, immediately after pressing <enter> when executing spawn\_tasks, why?). Experiment with taskShow, what do you see?**

Executing spawn\_tasks(50) and pressing (i+Enter) immediately before the printing function completes in order to capture running tasks which are printing.

**1. In the Host Shell, the following command and results**

**-> spawn\_tasks(50)**

value = 341811024 = 0x145f9f50

**-> i**

NAME ENTRY TID PRI STATUS PC ERRNO DELAY

---------- ------------ ---------- --- ---------- ---------- ---------- -----

tJobTask jobTask 0x14217a00 0 Pend 0x1015dbab 0x0 0

*….. there is information but not displayed here to save space….*

t81 printing 0x144ff958 90 Delay 0x101650a7 0x0 1

t82 printing 0x1455f060 90 Delay 0x101650a7 0x0 0

t83 printing 0x1455f370 90 Delay 0x101650a7 0x0 0

*….. there is information but not displayed here to save space….*

t101 printing 0x145f3620 90 Delay 0x101650a7 0x0 0

t102 printing 0x145f6ab8 90 Delay 0x101650a7 0x0 0

t103 printing 0x145f9f50 90 Delay 0x101650a7 0x0 0

value = 0 = 0x0

**2. In the Target Console, the following results are shown.**

*….. there is information but not displayed here to save space….*

I am task 341811024, iter: #47

I am task 341797560, iter: #48

I am task 341811024, iter: #48

I am task 341811024, iter: #49

**3. taskShow displays the content of Task Control Block**

taskShow TID, 0 returns the summary of the task

taskShow TID, 1 returns the details of the task

taskShow TID, 2 returns the list of all task

**-> taskShow 0x184a8fd0, 0**

NAME ENTRY TID PRI STATUS PC ERRNO DELAY

---------- ------------ ---------- --- ---------- ---------- ---------- -----

tWdbTask wdbTask 0x184a8fd0 3 Ready 0x1015dbab 0xb30008 0

value = 0 = 0x0

**A5. Control task execution using suspend, resume, delete (ts, tr, td) from the shell. Observe and comment on the produced output.**

Task execution (Suspend, Resume and Delete). Run the following command in shell:

**-> period (5, printing, 3)**

task spawned: id = 0x141f0b30, name = s44u5

value = 337578800 = 0x141f0b30

Target console will be printing the following repeatedly:

-> I am task 337578800, iter: #0

I am task 337578800, iter: #1

I am task 337578800, iter: #2

**Task Suspend**, use the following command on the Host Shell:

**-> ts(s44u5)**

value = 0 = 0x0

Target console will not show any more printing since the task is suspended.

**Task Resume**, use the following command on the Host Shell:

**-> tr(s44u5)**

value = 0 = 0x0

Target console will start printing again.

**Task Delete**, use the following command on the Host Shell:

**-> td(s44u5)**

value = 0 = 0x0

Target console will stop printing since the task is deleted.

**A6. Use the Debugger to execute spawn\_tasks(N). Experiment with attaching and detaching running tasks (printing), breakpoints and single-stepping. Refer to the Debugger on-line manuals. Comment on how easy/hard was it to use debugger.**

Answer: I do not feel confident answering this question because I do not understand how to do this section.

**A7. The tasks execute in such order that the one with less printing lines will be executing before the one with more iterations (see example below for a case with three tasks):**

I am task 73326808 <--- task 1

I am task 73225848 <--- task 2

I am task 73214976 <--- task 3

I am task 73225848 <--- task 2

I am task 73214976 <--- task 3

I am task 73214976 <--- task 3

**Modify the task priorities to reverse this order and to get printing as on the example below. Explain the source code changes. What instruction(s) must be changed and how? (Attach your modified source code with your deliverable)**

I am task 73214976 <--- task 3

I am task 73225848 <--- task 2

I am task 73326808 <--- task 1

I am task 73214976 <--- task 3

I am task 73225848 <--- task 2

I am task 73214976 <--- task 3

Answer: I do not feel confident answering this question because I do not understand how to do this section.

taskId = taskSpawn(NULL,90-i,0x100,2000,(FUNCPTR)printing,i,0,0,0,0,0,0,0,0,0);

**A8. Attempt use POSIX delay function nanosleep to replace taskDelay. Look in the documentation to find out what happens (the header file ‘time.h’ will need to be included). Attach modified source code with highlighted/commented new lines. Comment on the results of this experiment**

Answer: I do not feel confident answering this question because I do not understand how to do this section.

**Section 4: Observations, Comments, and Lessons Learned**

I have learned about WindRiver, few basic functions, the options and controls. What is actually happening and what is being done. There is more practice and exercise required in order to gain more experience and knowledge. However, this exercise was very helpful to understand many things that we are doing in labs and how system is behaving with different commands.

During the lab, I was only focused on completing the lab without error and not understanding the whole idea. This assignment helped understand some idea on what we are doing.

<http://www.vxdev.com/>

The above website has a lot of good material to understand different functions and commands. For example, SP, Repeat and Period was very easy to understand.