

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Operating System Principle |
| **Semester**: | 2nd semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | Ruoxuan Fu |
| **Student ID:** | 222019321062060 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Process Operations in Linux | | | |
| Date | | March, 2021 | Type | | √ Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements** 2. Learn the basic process operations in Linux 3. Review the writing, compiling and running of kernel modules 4. Understand process control block | | | | | |
| 1. **Experimental environment (**platform and software**)**   Virtualbox + Ubuntu (or other platform+linux system combinations) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results) 2. Tasks for this lab    1. Task 1   Experiment with the fork(), exec(), wait(), and exit() system calls for process operations in Linux.   * + 1. In the main process, create a child process     2. Load a new task in the child process     3. The main process waits for the child to exit, and accesses the child’s exit status     4. The status accessed by the main process should be 255   1. Task 2   Use kernel module to access the two values jiffies and HZ defined in the linux kernel:   * HZ: the frequency of timer interrupt * jiffies: the number of timer interrupt since system boot   Please output the value of jiffies twice, i.e. when the module is loaded and when the module is removed. Then based on the two jiffies and HZ, compute how long your kernel module stays in the kernel.   * 1. Task 3   Write, compile, and load a Linux kernel module to travese through the list of PCBs in the kernel, and   * + 1. Output to the kernel buffer the PID, state, and the executable name of each process     2. Count the number of processes in your system     3. Compare the results in i) and ii) with the list given by the ‘ps -el’ command  1. Please provide your procedure and source codes to perform the tasks.   Procedure:  Task1:  1. write a source file to create a parent process and a child process which are aimed to interact with execl(),wait() and exit()  task1.c:   1. #include <unistd.h> 2. #include <stdio.h> 3. #include <stdlib.h> 4. #include <sys/wait.h> 6. **int** main() 7. { 8. pid\_t new\_process\_id; 10. new\_process\_id = fork(); 12. **int** status; 14. **if**(new\_process\_id ==0) 15. { 16. printf("This is in child process: %d\n", getpid()); 17. execl("./mysum", " ","2","3",NULL); 18. printf("This is the rest of the cild process"); 19. } 20. **else** { 21. printf("This is in parent process: %d\n", getpid()); 22. wait(&status); 23. printf("child exit status: %d\n", WEXITSTATUS(status)); 24. } 26. **return** 0; 27. }   mysum.c:   1. #include <stdio.h> 2. #include <stdlib.h> 4. **int** main(**int** argc, **char** \*argv[]) 5. { 6. **if**(argc != 3) 7. { 8. printf("Please input two integers as the arguments!\n"); 9. exit(-1); 10. } 11. printf("The sum is: %d\n", atoi(argv[1]) + atoi(argv[2])); 13. exit(-1); 14. }   2. First I should compile mysum.c to a binary file and then compile the task1.c to a executable file and run it.    3. Result:    Task2.c：  1. The kernel module needs to write a source file of c. There are two functions in this source file, one executed when the module is installed and one executed when the module is uninstalled. So you need to get the value of jiffies at install and uninstall time respectively. The difference between the two jiffies and the hz is used to obtain the time of existence of the module   1. #include <linux/module.h> 2. #include <linux/kernel.h> 3. #include <linux/param.h> 5. **int** TimeInterval = 0; 6. **int** AccessJiffies\_entry(**void**) 7. { 8. TimeInterval = jiffies - 0; 9. printk(KERN\_INFO "\"jiffies\" is %ld\n", jiffies); 10. **return** 0; 11. } 13. **void** AccessJiffies\_exit(**void**) 14. { 15. TimeInterval = jiffies - TimeInterval; 16. printk(KERN\_INFO "\"jiffies\" is %ld and the existing time of this module is %d second\n",jiffies,TimeInterval/HZ); 17. } 19. module\_init(AccessJiffies\_entry); 20. module\_exit(AccessJiffies\_exit); 22. MODULE\_LICENSE("GPL"); 23. MODULE\_DESCRIPTION("Access jiffies"); 24. MODULE\_AUTHOR("Fu Ruoxuan");   2. In makefile file, we should point out the ko file that we want to create   1. obj-m += AccessJiffies.o 3. all: 4. make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) modules 5. clean: 6. make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) clean   3. result | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   **Analysis of result:**  In the results of the first experiment, the code in mysum successfully overwrites the remaining code in the child process, and the child process exits in mysum returning -1.  In the results of the last experiment, we can clearly see that the jiffies values and the existing time of this module.  **Harvest:**  In exit(-1), -1 is returned, but what you get in WEXITSTATUS() is the lower eight bits of the returned value, which is the binary complement of -1. And when converting it to decimal, he becomes 255.  The Linux kernel issues a timer interrupt (IRQ 0) every fixed period, and HZ is used to define how many timer interrupts per second. for example, a HZ of 1000 means there are 1000 timer interrupts per second.  Jiffies is a Linux kernel variable (32-bit variable, unsigned long), which is used to record how many ticks have been passed since the system was booted. every time a timer interrupt occurs, the Jiffies variable is added by one.  In linux systems, the printf function is a row-buffered output that flushes the contents of the buffer to standard output (stdout) when printf is encountered, or when the buffer is full.  Therefore, printf ("p"); The display of such statements cannot be immediately displayed on the screen, but printf ("p\n") will print correctly. In order to solve this problem can be achieved: fflush (stdout); Solution, which becomes:  printf("p");  fflush(stdout);  The problem can be solved.  **Existing problem:**  None. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |