

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Operating System Principle |
| **Semester**: | 2nd semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | 吴嘉诚 |
| **Student ID:** | 222019321062111 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Process Operations in Linux | | | |
| Date | | April, 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 5. Use kernel module to access 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 child exits with status 255   1. Task 2  1. Find the sched.h file containing the definition of process control block (task\_struct) on your virtual machine 2. Locate the starting line and the ending line of task\_struct 3. Locate the lines that define process identifier, process state, as well as and the process’s executable name    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 (hint: useful command **wc -l**)  1. Please provide your procedure and source codes to perform the tasks.   **Task1:**   1. Refers to previous files(myfork.c, mywaitexit.c, mysum.c)   And write code:  #include <stdio.h>  #include <stdlib.h>  #include <sys/wait.h>  #include <unistd.h>  int main(int argc, char const \*argv[])  {  pid\_t new\_pid;  int status;  new\_pid = fork();  if(new\_pid == 0)  {  printf("This is in child process: %d, now execute mysum...\n",getpid());  execl("./mysum", " ", "6", "8",NULL);  //the rest will never be executed, as the child process has been replaced by mysum  printf("This is the rest of the child process\n");  exit(255);  }  else  {  printf("This is parent process: %d, and it is going to wait...\n",getpid());  wait(&status);  printf("The exit status of the child process is: %d\n", WEXITSTATUS(status));  }  return 0;  }   1. Compile and execute   2021-04-09 22-42-04 的屏幕截图   1. Troubleshooting   At the very beginning, i just write codes above without considering the file: mysum. So i can’t get the correct exit status of child process because the ‘mysum’ process has taken the place of the original child process. So i need to modify the mysum.c like this:  #include<stdio.h>  #include<stdlib.h>  int main(int argc, char const \*argv[])  {  if (argc != 3)  {  printf("please input two integers as arguments!\n");  }  else  {  printf("the sum is: %d\n", atoi(argv[1])+atoi(argv[2]));  }  exit(255);//used in task1  return 0;  }  As a result, the parent process can get this 255 exit status from mysum.  **Task2:**   1. Use ‘find’ command to locate the .h file   2021-04-10 14-42-38 的屏幕截图   1. Open the file and find task\_struct, start and end line:   2021-04-09 21-41-08 的屏幕截图  2021-04-09 21-46-15 的屏幕截图   1. Definitions of process identifier, process state, executable name   2021-04-10 15-20-21 的屏幕截图  2021-04-10 15-15-46 的屏幕截图  2021-04-10 15-26-24 的屏幕截图  **Task3:**   1. Write codes:   #include <linux/module.h>  #include <linux/kernel.h>  #include <linux/sched/signal.h> //for\_each\_process()  int t3\_entry(void)  {  struct task\_struct\* curr;  int count = 0;  for\_each\_process(curr)  {  printk(KERN\_INFO "PID: %d, Name: %s, state: %ld\n",curr->pid,curr->comm,curr->state);  count++;  }  printk(KERN\_INFO "Total: %d processes\n",count);  return 0;  }  void t3\_exit(void)  {  printk(KERN\_INFO "exit...");  }  module\_init(t3\_entry);  module\_exit(t3\_exit);  MODULE\_LICENSE("GPL");  MODULE\_DESCRIPTION("kernel module test: ps");  MODULE\_AUTHOR("zhj,");   1. make and insmod   2021-04-11 16-42-17 的屏幕截图   1. dmesg and ps -el | wc -l   2021-04-11 16-37-11 的屏幕截图 | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   **Task1:**   1. **I learned about exec(), wait() and exit status.** 2. **I realized exec() is actually take the place of the current process. The rest of the original program will never be executed.**   **Task2:**   1. **I learned how to use “find” cmd and the shortcut for searching chars in text editor.** 2. **Finally found these variables in sched.h, and known their name.**   **Task3:**   1. **At first i don’t know why task2 needs to find those variables in the header file, now in task3 i understand it. We need to use them in this task by calling their name!** 2. **But when comparing the total output with ps command, i found that the ps always get 1 more than my own output. I believe it’s because the ps itself is a process!**   **2021-04-11 16-55-12 的屏幕截图** | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |