

**Lab report**

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

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | C Programming, Makefile and Linux Kernel Module | | | |
| Date | | March 15, 2021 | Type | | √ Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Learn to do C programming with Linux    2. Learn how to write simple Makefile for managing C projects    3. Learn how to write, compile, and load linux kernel modules | | | | | |
| 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   Write, compile, and run a C program with at least two \*.c source files and one \*.h head file.   * 1. Task 2   Use Makefile and the make tool to compile your C program with at least two \*.c source files and one \*.h head file as in Task 1.   * 1. Task 3   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. Please provide your procedure and source codes to perform the tasks. 2. Task 1   ① Write two \*.c source files and one \*.h head file        ② Complie these sources      ③ Link these obj files and library files      ④ Run the program     1. Task2   ① Write a Makefile file  ② Run the Makefile file | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Analysis of results：  The experiment was able to obtain the desired results. After gcc compiling, run the .out file to get the correct result. And using the makefile file can also be compiled correctly and get the correct result  Harvest:  The gcc compilation process consists of four operations, preprocessing, compilation, assembly, and linking. The command ”gcc -e \*.c”’s function is to process the preprocessing instructions. The command ”gcc -s \*.i”’s function is to process the program after preprocessing and convert it to assemble file. The command ”gcc -c \*.s”’s function is to process the assemble file and convert it to BINary files. The command ”gcc \*.o”’s function is to Use the linker to link this target file with other target files, library files, startup files, etc. to generate executable files. Additional target files include statically linked libraries and dynamically linked libraries.  The Makefile file describes the compilation rules for C/C++ projects on Linux systems and is used to automate the compilation of C/C++ projects. Once the Makefile file is written, a single make command is all that is needed to start compiling the entire project automatically, eliminating the need to execute GCC commands manually.  In makefile, we should pay attention to the indentation of commands. At the beginning of the command we should use tabs for indentation.  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: | | | | |