**深圳大学实验报告**

**课程名称： 计算机系统(2)**

**实验项目名称： 链接实验**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 罗秋明**

**报告人： 刘俊楠 学号： 2017303010 班级： 计科01**

**实验时间： 2021 年 6 月 9 日**

**实验报告提交时间： 2021 年 6 月 19 日**

**教务处制**

|  |
| --- |
| **一、实验目的与要求：**   1. 了解GNU make、掌握makefile文件的编写 2. 了解华为鲲鹏及X86在C/C++类编译类语言中的重要差异 3. 掌握静态链接与动态链接 4. 理解系统加载可执行程序所需的处理过程。 |
| **二、实验内容：**   * 1. 部署准备工作，购买并创建华为云ARM（鲲鹏）Linux弹性云服务器，配置后登陆弹性云服务器。   （1）登陆华为云，进入控制台    （2）选择华北四Region，然后点击弹性云服务器    （3）点击购买弹性云服务器  （4）配置如下             * 1. **本地已经安装用于连接 Linux 弹性云服务器的工具，如Putty**   **1.3 登陆弹性云服务器**  登陆弹性云服务器(图1-3-1、图1-3-2)，直接通过小桌面Terminal登陆ECS服务器，打开ESC命令行环境，如下图所示：（账号密码系统会自动分配）  说明：一般情况下，我们都是使用root账户来进行登陆。用户名为：root，密码为登陆密码;  输入密码时，命令行窗口不会显示密码，输完之后直接回车即可。    图1-3-1    图1-3-2   * 1. 修复程序Bug并编写makefile完成程序构建。   2. 创建静态链接库libmymath.a，并调用该静态库。   3. 创建动态链接库libmymath.so，并调用该动态库。   4. 使用gdb跟踪分析一个execve系统调用内核处理函数sys\_execve。 |
| **三、实验结果及分析：**   * **1、**部署准备工作，进入华为云官网，购买并创建华为云ARM（鲲鹏）Linux弹性云服务器，配置后登陆弹性云服务器。 * **主要思路：**   1-1：根据实验提示购买并创建华为云ARM（鲲鹏）Linux弹性云服务器    **图1-1弹性云服务器**  1-2：配置后登录弹性云服务器    **图1-2弹性云服务器shell界面**  1-3：通过Putty登录弹性云服务器    **图1-3 Putty登录弹性云服务器**  1-4：使用pscp传输文件至服务器    **图1-4 使用pscp传输文件**   * **内容二：**修复程序Bug并编写makefile完成程序构建 * **主要思路：**   2-1：修复main程序的bug，对于判断语句if 和else if中是通过判断char 型变量flag的正负来实现的，查阅文档后得知（图2-2），在x86中char是unsigned，在鲲鹏中是signed，因此对于鲲鹏平台，char类型的flag应该修改为：**signed char flag**（图2-1）    **图2-1 main.c代码的bug**    **图2-2 char在不同架构下的差异**  2-2：编写makefile文件，需要注意的是由于要调用数学库，需要在.o文件后加上-lm才能编译正确    **图2-2构建makefile文件**  2-3：使用make命令，文件夹会生成一个可执行文件edit    **图2-3 make命令生成可执行文件**  2-4：进行测试样例进行验证，发现答案都是正确的。    **图2-4验证测试样例**   * **内容三：创建静态链接库libmymath.a，并调用该静态库** * **主要思路：** * **对于链接静态库，实质上相当于是把一堆.o文件打了个包，当你需要用到这里面的函数时，把这个静态库文件加到链接的输入里。优点是比较简单，缺点是每次链接的时候都要把一大堆.o文件拷贝过来，占用大量存储器空间，而且当需要改变静态库里某个.o文件时要把这一大堆重新打包。**   3-1：通过make clean指令删除之前的.o文件。  将所有指定的源文件，都编译成相应的目标文件：使用gcc -c的方式重新生成.o文件。  然后使用 ar 压缩指令，将生成的目标文件打包成静态链接库，其基本格式如下：其中 ar cr libmymath.a add.o sub.o 为将两个.o文件打包成静态库mymath。（图3-1）  （库文件的命名规范是以lib开头（前缀），紧接着是静态库名，以 .a为后缀名。）    **图3-1生成静态库**  3-2：对生成的可执行文件进行测试，可见与makefile生成的文件mainexe运行出来的结果是一样的,答案正确。（图3-2）  其中核心代码为gcc -o edit1 -lm main.c variance.c –L. -lmymath。其中-L后面指定静态库的文件路径(当前文件夹)。-lm是由于使用到了数学库的函数。    **图3-2验证答案**   * **内容四：创建动态链接库libmymath.so，并调用该动态库** * **主要思路：**   4-1：在系统提示符下键入以下命令得到动态库文件libmymath.so(图4-1)：  gcc -fPIC -shared -o libmymath.so add.c sub.c以创建动态链接库libmymath.so  PIC: 在编译的时候，我们需要告诉编译器，这些对象文件是用来做动态链接库的，所以要用地址不无关代码（Position Independent Code (PIC))。    **图4-1创建动态库mymath.so**  4-2：用file命令确认是否是共享库   * 通过使用指令file libmymath.so命令发现确实是共享库，且为ARM下面的动态库。而对比之下libmymath.a并不是共享库（图4-2）。     **图4-2确认共享库**  4-3：动态链接库分别采取默认搜索路径和修改环境变量LD\_LIBRARY\_PATH两种方式   * Linux动态链接库的搜索路径按**优先级**排序：   1.编译目标代码时指定的动态库搜索路径（在编译时通过gcc 的参数”-Wl,-rpath,”指定）。  2.环境变量LD\_LIBRARY\_PATH指定的动态库搜索路径；  3.配置文件/etc/ld.so.conf中指定的动态库搜索路径；  在/etc/ld.so.conf的第一行有个引用命令：include ld.so.conf.d/\*.conf  因此，最好是在ld.so.conf.d目录下创建一个你的程序依赖的配置文件，配置文件内容为程序依赖的动态链接库的路径，一个路径一行。当指定多个动态库搜索路径时，路径之间用冒号”：”分隔。  注意：在添加完配置文件后执行ldconfig使其生效。  4.默认的动态库搜索路径/lib；  5.默认的动态库搜索路径/usr/lib；   * 若直接进行gcc编译，不指明动态库链接路径，则编译失败     **图4-3未修改环境变量导致编译失败**   * 修改环境变量LD\_LIBRARY\_PATH，需要指定当前文件夹的路径LD\_LIBRARY\_PATH内的路径。     **图4-4修改环境变量**   * 验证答案，编译程序成功并运行结果     **图4-5验证答案**  4-4：如何实施可执行程序装载时动态链接和运行时动态链接？   * **装载时动态链接：**在装入一个目标程序模块时，若发生一个外部模块调用，将引起动态链接程序去找出相应的外部目标模块，装入内存，修改目标程序模块中的相对地址。其是分别载入，当把一个模块载入内存时检查有调用关系的模块载入，比静态链接节省了许多开销。 * **运行时动态链接**：在执行过程中，若发现一个被调用模块未装入内存时，通过调用接口找到该模块，将其装入内存，并连接到调用者模块上。其是把当前模块调用的模块推迟到调用的时候再载入。 * 当应用程序调用LoadLibrary 或 LoadLibraryEx 函数时，系统就会尝试按载入时动态链接搜寻次序（参见载入时动态链接）定位DLL。如果找到，系统就把DLL模块映射到进程的虚地址空间中，并增加引用计数。如果调用LoadLibrary或LoadLibraryEx 时指定的DLL其代码已经映射到调用进程的虚地址空间，函数就会仅返回DLL的句柄并增加DLL引用计数。 * 运行时动态链接的实现需要包含dlfcn.h头文件，以及其中的dlopen(),dlsym(),dlclose()函数（图4-6）。     **图4-5实现运行时动态链接代码设计**    **图4-6实现运行时所需函数**  4-5：用ldd工具确认某可执行程序是否引用了该动态库。    **图4-7确认可执行程序引用动态库**   * **内容五：使用gdb跟踪分析一个execve系统调用内核处理函数sys\_execve** * **主要思路：** * **通过gdb调试内核函数sys\_execve后，分析如下** * 通过gdb-silent模式进入gdb，然后重新打开一个gdb进入内核，并执行目标程序，对execve系统调用设置断点，并且开始调试。 * 新的可执行程序通过修改内核堆栈eip作为新程序的起点，从new\_ip开始执行后start\_thread把返回到用户态的位置从main函数位置的下一条指令变成新加载的可执行文件的入口位置。 * 当执行到execve系统调用时，进入内核态，用execve()加载的可执行文件覆盖当前进程的可执行程序。 * 当execve系统调用返回时，返回新的可执行程序的执行起点（main函数），所以execve系统调用返回后新的可执行程序能顺利执行。 * execve系统调用返回时：  1. 如果是静态链接，elf\_entry指向可执行文件规定的头部 2. 如果需要依赖动态链接库，elf\_entry指向动态链接器的起点。其中动态链接主要是由动态链接器ld来完成的。   **四、实验总结与体会：**   * 通过本次实验，我明白了Makefile的语法以及如何编写Makefile，并且对鲲鹏的ARM架构与x86的架构有了更加清晰的区别。 * 本实验对于程序编译，以及运行时链接、静态链接、动态链接等等有了更加深入的探讨，让我对程序产生的过程有了清晰的理解。 * 通过GDB调试深入了解鲲鹏内核里的系统调用exccve是如何处理与链接可执行程序。 |
|  |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2021年 月 日 |
| 备注： |