Linux 内核云上实验指导(一)

2020年4月

华为鲲鹏云服务器

使用华为云账号登陆华为云平台，统一注册的账号包含200元华为云代金券。

创建虚拟私有云(VPC)

1. 选择`产品 -> 基础服务 -> 虚拟私有云 VPC -> 创建VPC -> 访问控制台 -> 创建虚拟私有云`
2. 区域选择`华东-上海一`，其他默认即可
3. `访问控制 -> 安全组`中可以设置安全组。默认已放通22端口和3389端口，如有需要，可以添加规则->全部放通。

购买弹性云服务器(ECS)

1. 选择`产品 -> 基础服务 -> 弹性云服务器 ECS -> 立即购买`
2. 基础配置

注：实验与华为鲲鹏云合作，ECS规格要求选择基于arm架构的鲲鹏服务器，不要选成x86服务器。

计费模式	区域	规格	镜像	系统盘
按需计费	华东-上海一	鲲鹏计算 2vCPUs | 4GB	Ubuntu 18.04	至少40GB

3. 网络配置

网络	安全组	弹性公网IP
之前创建的VPC	之前创建的安全组	现在购买

4. 高级配置

设置云服务器名称（e.g., ecs-zhangsan）、用户名（e.g., root）、密码（e.g., LinuxKernel2020）

云备份选择<暂不购买>

5. 确认配置

务必确认收费模式是否为<按需计费>，注意每小时金额。若误操作可能会导致账户余额不够、无法完成实验。

下载SSH工具

推荐 mobaxterm (windows)、 vscode + ssh remote插件、Royal TSX (mac)

当然，mac OS的terminal自带ssh功能

登陆ECS

创建好ECS后，可以在`控制台 -> 弹性云服务器(区域务必选择上海一)`中看到弹性公网IP。

使用SSH工具，输入EIP、用户名和密码，或`ssh usr@EIP`即可登陆。

创建私有镜像

控制台 - 云服务器控制台 - 镜像服务 - 创建私有镜像

保存私有镜像可以方便重新购买ECS时恢复之前搭好的环境。如有需要，可以在删除ECS之前制作私有镜像。

实验零 编译内核

实验要求

将ECS的内核版本更新到5.5以上。

注：Ubuntu 18.04操作系统默认的内核版本为4.15，可通过`uname -a`查看。需要将其内核版本更新为5.5以上，以5.5.9为例。

下载linux kernel源码

登陆ECS，到linux kernel官网 下载最新源码包。

```
$wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.5.x.tar.xz
```

注：如果ECS的下载速度太慢，也可以先下载到本地，再从本地上传到ECS(参考：scp命令、sftp工具)。

解压

```
tar xvf linux-5.5.x.tar.xz
cp -r linux-5.5.x /usr/src/
cd /usr/src/linux-5.5.x
```

更换apt源

ubuntu自带的apt源经常出现连接不上的情况，最好换成国内源，这里以华为源为例。

```
cp /etc/apt/sources.list /etc/apt/sources.list.bak
rm /etc/apt/sources.list
lsb_release -c
vim /etc/apt/sources.list     //注：x86对应的是ubuntu，arm对应的是ubuntu-ports
```

```
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic main restricted
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic main restricted
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-updates main restricted
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-updates main
restricted
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic universe
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic universe
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-updates universe
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-updates universe
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic multiverse
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic multiverse
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-updates multiverse
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-updates multiverse
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-backports main
restricted universe multiverse
```

```
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-backports main
restricted universe multiverse
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-security main restricted
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-security main
restricted
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-security universe
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-security universe
deb http://mirrors.huaweicloud.com/ubuntu-ports bionic-security multiverse
deb-src http://mirrors.huaweicloud.com/ubuntu-ports bionic-security multiverse
```

安装依赖

```
apt-get update
apt-get install gcc make libncurses5-dev openssl libssl-dev build-essential
pkg-config libc6-dev bison flex libelf-dev
```

配置内核

```
cd /usr/src/linux-5.5.x
//make clean
//make mrproper
//make menuconfig                                   ->图形界面
//make defconfig                                    ->根据主机架构自动选择对应arch下的默
认配置
cp /boot/config-4.15.0-70-generic .config       ->涉及虚拟硬盘、虚拟网卡等配置 默认
的没有打开
vim .config   //VIRTIO_GPU  ->  #VIRTIO_GPU     ->没有GPU，所以不要编译这个模块
make olddefconfig                                 ->按照.config来配置
```

编译和模块安装

```
//tmux new-session -s kernel        ->防止make过程中ssh断开
make -j2                            -> -jn表示用n个进程，n一般设置为core的数量
make modules_install                ->安装模块
make install                        ->会自动安装kernel、创建initramfs、更新引导
```

注：编译内核的过程可能比较久，期间如果因为网络原因断开了与服务器的连接，可能会功亏一篑。为了避免这种情况，这里推荐使用tmux工具，具体用途和命令可自行google。

重启

```
shutdown -r now
```

检查内核版本

```
uname -a
```

实验一 模块编程

实验要求

编写四个模块，分别实现以下功能：

1. 模块一，加载和卸载模块时在系统日志输出信息
2. 模块二，支持整型、字符串、数组参数，加载时读入并打印
3. 模块三，在/proc下创建只读文件
4. 模块四，在/proc下创建文件夹，并创建一个可读可写的文件

实验提示

1. 模块可以写在一个c文件中，模块参数传递参考宏定义`module_param(name, type, perm)`，需要用到头文件`linux/moduleparam.h`。
2. 编写Makefile文件将c源码编译成.ko的模块。
3. 模块下proc目录和文件的创建参考`proc_mkdir`和`proc_create`函数。

实现思路（仅供参考）

```
//<Makefile>
可参考PPT
```

```
//<module1.c>
//Test for installing and removing of module.
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
static int __init hello_init(void){
    //TODO: 加载模块时printk输出信息
}

static void __exit hello_exit(void){
    //TODO: 卸载模块时printk输出信息
}
module_init(hello_init);
module_exit(hello_exit);
```

```
//<module2.c>
//Support for int&str&array parameter
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/kernel.h>

//TODO: 声明用于接收参数的变量

//TODO: 使用module_param来指定模块可接受的参数
```

```
static int __init hello_init(void)
{
    //TODO: 加载模块时printk输出信息，打印参数的值
}
static void __exit hello_exit(void)
{
    //TODO: 卸载模块时printk输出信息
}
module_init(hello_init);
module_exit(hello_exit);
```

```
//<module3.c>
//read-only proc file
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>

static int hello_proc_open(struct inode *inode, struct  file *file) {
    //定义文件操作
}

static const struct file_operations hello_proc_fops = {
    //TODO: 指定文件操作
};

static int __init hello_proc_init(void) {
    //TODO: 加载模块时printk输出信息
}

static void __exit hello_proc_exit(void) {
    //TODO: 卸载模块时printk输出信息，删除创建的proc文件
}

module_init(hello_proc_init);
module_exit(hello_proc_exit);
```

```
//<module4.c>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>

#include <linux/fs.h>          // for basic filesystem
#include <linux/proc_fs.h>     // for the proc filesystem
#include <linux/seq_file.h>    // for sequence files
#include <linux/slab.h>        // for kzalloc, kfree
```

```c
#include <linux/uaccess.h>    // for copy_from_user
#define BUF_SIZE 10

//TODO: 可以在这里声明需要的变量，注意static关键字的使用

// file_operations -> write
static ssize_t hello_write(struct file *file, const char __user *buffer, size_t
count, loff_t *f_pos)
{
    //TODO: 定义文件操作
}

// seq_operations -> open
static int hello_open(struct inode *inode, struct file *file)
{
    //TODO: 定义文件操作
}

static const struct file_operations hello_fops =
{
    //TODO: 指定文件操作
};

// module init
static int __init hello_init(void)
{
    //TODO: 创建proc目录、创建proc文件
}

// module exit
static void __exit hello_exit(void)
{
    //TODO: 删除proc目录、删除proc文件，如有手动分配内存空间，记得释放
}

module_init(hello_init);
module_exit(hello_exit);
```

```
# 实验效果测试可能用到的指令
make
sudo insmod mod.ko
dmesg
sudo insmod module2.ko int_var=666 str_var=hello int_array=10,20,30,40
dmesg
sudo insmod module3.ko
cat /proc/hello_proc
sudo insmod module4.ko
cat /proc/hello_dir/hello
echo nice > /proc/hello_dir/hello
cat /proc/hello_dir/hello
```

注：以上模板仅供参考，也可以完全按照自己的思路来实现，实验报告中如果能清晰具体地写出自己的实验思路和具体实现会比较有亮点。

作业提交

提交渠道：Canvas

提交文件："学号_姓名_project1.zip"

- 源码文件夹"学号_姓名_project1_src"（*.c Makefile）
- 实验报告《学号_姓名_project1_report.pdf》，包括但不限于实验过程、实验效果截图、实验心得。