


具体内容如下 2-1, 2-2, 2-3:

作业2-1: makefile工程管理器的使用

- 内容: 编写一个包含多文件的akefile
- 目的: 通过对包含多文件的Makefile 的编写, 熟悉各种形式的Makefile, 并且进一步加深对Makefile 中用户自定义变量、自动变量及预定义变量的理解
- 平台: PC机, Ubuntu操作系统, gcc等工具
- 具体:
 - 1. 用vi在同一目录下编辑两个简单的hello程序,
 - 2. 仍在同一目录下用vi 编辑 Makefile, 不使用变量替换, 用一个目标体实现 (即直接将hello.c 和hello.h 编译成 hello 目标体)。并用make 验证所编写的Makefile 是否正确。
 - 3. 将上述 Makefile 使用变量替换实现。同样用make 验证所编写的 Makefile 是否正确
 - 4. 编辑另一 Makefile, 取名为Makefile1, 不使用变量替换, 但用两个目标体实现 (也就是首先将hello.c 和 hello.h 编译为 hello.o, 再将 hello.o 编译为 hello), 再用 make 的 ‘-f’ 选项验证这个Makefile1 的正确性。
 - 5. 将上述 Makefile1 使用变量替换实现



```
#hello.c
#include "hello.h"
int main()
{
    printf("Hello everyone!\n");
}

#hello.h
#include <stdio.h>
```

作业2-2: GDB调试工具的使用

- 内容: 将原来出错的程序经过gdb调试, 找出bug, 修改源码, 输出正确的倒序显示字符串的结果
- 目的: 通过调试一个有问题的程序, 进一步熟练使用操作, 熟练使用gcc 编译命令及gdb 的调是命令, 通过对有问题程序的跟踪调试, 进一步提高发现问题和解决问题的能力
- 平台: 带有linux操作系统的PC机
- 步骤:
 - 1. 使用 **vi 编辑器**, 生成greet.c 的文件。此代码的原意为输出倒序输出main函数中定义的字符串, 但结果显示没有输出
 - 2. 使用 **gcc 编译**这段代码
 - 3. 运行生成的可执行文件greet, 观察运行结果。
 - 4. 使用 **gdb 调试**程序, 通过设置断点、单步跟踪, 一步步找出错误所在。
 - 5. 纠正错误, 更改源程序并得到正确的结果。

作业2-2: GDB调试工具的使用

下述调试步骤可供参考，可根据个人思路调整：

启动 gdb 调试: `gdb greet`

- 查看源代码
- 在for 循环处设置断点
- 在printf 函数处设置断点
- 查看断点设置情况
- 运行代码
- 单步运行代码
- 查看暂停点变量值
- 继续单步运行代码数次，并使用命令查看出错语句
- 退出 gdb
- 重新编辑 `greet.c`
- 重新编译

`greet.c`

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int display1(char *string);
int display2(char *string);

int main (int argc, char **argv)
{
    char string[] = "Embedded Linux";
    display1 (string);
    display2 (string);
    return 0;
}

int display1 (char *string)
{
    printf ("The original string is %s \n", string);
}
```

```
int display2 (char *string1)
{
    char *string2;
    int size,i;
    size = strlen (string1);
    string2 = (char *) malloc (size+1);
    for (i = 0; i < size; i++)
        string2[size - i] = string1[i];
    string2[size+1] = '\0';
    printf ("The string afterward is %s\n", string2);
    free(string2);
}
```

```
correct? error: ld returned 1 exit status
linux@linux-pc:~/workdir/fs3399/application/test$ vim greet.c
linux@linux-pc:~/workdir/fs3399/application/test$ gcc -g greet.c -o greet
linux@linux-pc:~/workdir/fs3399/application/test$ ./greet
The original string is Embedded Linux
The string afterward is
```

作业2-3: Linux 系统 UDP 网络协议编程

- 目的: 熟悉socket网络编程的基本方法
- 平台: Vmware虚拟机, Ubuntu操作系统, gcc编译器
- 内容: 通过一个简单的udp服务器端, 接收客户端的连接请求, 并接受客户端发来的信息
- 具体步骤:
 - 建立目录 `$ mkdir ~/workdir/linux/application/udp -p`
 - 将client.c 和 server.c 拷贝到上述文件夹下
 - 分别编译c文件, 生成可执行文件client 和 server
 - 执行文件: `./server 192.168.100.240 8888` (各自ubuntu系统的IP可通过ifconfig查看)

```
linux@linux-pc:~/workdir/linux/application/tcp$ ifconfig
ens33: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.100.240 netmask 255.255.255.0 broadcast 192.168.100.255
    ether 00:0c:29:97:01:19 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- 按下ctrl+Alt+t, 打开另一个终端, 进入该目录, 运行服务端程序
- `./client 192.168.100.240 8888`
- 提示符下输入任意字符串, 如: hello
- 查看服务器终端是否能接收该信息并显示

作业2-3: Linux 系统 UDP 网络协议编程

- 作业要求:
 - 根据具体内容, 完成客户端向服务器发送信息的要求; 可尝试发送不同的字符串信息。
 - 剖析代码client.c 和 server.c, 加入注释。
 - 参考:
 - 网络编程中的Socket 该怎么理解? - 知乎
<https://www.zhihu.com/question/638114381>
 - UDP 协议 (zhihu.com)

作业2-4: linux系统多进程实验

- 将【华清远见嵌入式 ARM 实验箱资料\程序源码\Linux 应用实验源码】目录拷贝到虚拟机共享目录【D:\share\】下
- 将 09. Linux 系统 fork 等系统调用实验实验部分代码拷贝到虚拟机linux下

```
1. 建立相关目录:
$ mkdir ~/workdir/linux/application/fork_test -p
$ cd ~/workdir/linux/application/fork_test

linux@ubuntu64-vm:~$
linux@ubuntu64-vm:~$ mkdir ~/workdir/linux/application/fork_test -p
linux@ubuntu64-vm:~$ cd ~/workdir/linux/application/fork_test
linux@ubuntu64-vm:~/workdir/linux/application/fork_test$ ls

图 11-1 建立目录

2. 将代码从共享目录拷贝入虚拟机 Linux 操作系统下: (可使用【Ctrl+空格】切换输入法)
$ cp /mnt/hgfs/share/Linux 应用实验源码/09. Linux 系统 fork 等系统调用实验/实验代码/93.c .
$ ls

linux@ubuntu64-vm:~/workdir/linux/application/fork_test$ cp /mnt/hgfs/share/Linux应用实验源码/09.
Linux系统fork等系统调用实验/实验代码/93.c .
linux@ubuntu64-vm:~/workdir/linux/application/fork_test$ ls
93.c

图 11-2 拷贝代码

3. 执行代码
$ gcc 93.c -o fork
$ ./fork //执行观察结果
```

作业2-4: linux系统多进程实验

- fork 后父子进程会同步运行，但父子进程的返回顺序是不确定的。设两个变量 **global** 和 **test** 来检测父子进程共享资源的情况
- 通过观察运行结果，跟踪 **global** 和 **test** 变量，体会 linux 多进程调度的一些机制。

93.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

int global=22;
char buf[]="the test content\n";

int main(void)
{
    int test=0,stat;
    pid_t pid;
    if(write(STDOUT_FILENO, buf, sizeof(buf)) != sizeof(buf))
    { perror("write error"); }
    printf(" fork test!\n");
    /* fork */
    pid = fork(); /*we should check the error*/
    if (pid == -1)
    {
        perror("fork");
        exit(0);
    }
    else if (pid == 0)
    {
        global++; test++;
        printf("global=%d test=%d Child,my PID is %d\n",global,test,getpid());
        exit(0);
    }
    /*else be the parent*/
    global+=2;
    test+=2;
    printf("global=%d test=%d Parent,my PID is %d\n",global,test,getpid());
    exit(0);
    //printf("global=%d test=%d Parent,my PID is %d",global,test,getpid());
    //_exit(0);
}
```

作业提交:

- 作业命名: 姓名+学号+2.zip (任意压缩软件)
 - 包含:
 - pdf 文档 1 份, 命名: 作业 2.pdf
 - 包含: 作业 2-1 中的实操过程截屏说明部分+作业 2-2 中的说明文档部分 (阐述清楚查找 bug 的调试过程并附必要截屏) +作业 2-3 中操作过程截屏部分+作业 2-4 中实操过程截屏+实验结果分析和体会
 - Makefile 文件 , Makefile1 文件
 - greet.c 源码 1 份
 - 注释后的 client.c、server.c 和 93.c, 分别命名为:
client_mark.c , server_mark.c, 93_mark.c
- 提交时间: 2025.10.19 晚 12 点
- 提交路径 : course.xmu.edu.cn