

**实验（践）报告**

**2024-2025学年第1学期**

|  |  |
| --- | --- |
| **课程名称** | 操作系统(Linux) |
| **学 院** | 远景学院 |
| **专 业** | 计算机科学与技术 |
| **班 级** | 2022级1班 |
| **姓 名** | 唐诗益 |
| **学 号** | 2022210901 |
| **成 绩** |  |
| **任课教师** | 邓祥凯 |

**重庆移通学院计算机学院（部）**

|  |  |  |  |
| --- | --- | --- | --- |
| **实验（践）报告评阅记录表** | | | |
| 课程编号 | 020348 | 课程名称 | 操作系统（Linux） |
| **实验（践）项目/任务** | | **分值** | **得分** |
| 1.Linux系统安装 | | 100 |  |
| 2.Linux常用命令 | | 100 |  |
| 3.Linux系统C程序的编写 | | 100 |  |
| 4.Linux进程管理 | | 100 |  |
| 5.进程调度算法模拟 | | 100 |  |
| 6.银行家算法 | | 100 |  |
| **成绩** | | |  |
| **教师签字** | | |  |

备注：成绩为各实验（践）项目/任务得分的平均值。

目录

[第1次实验 Linux系统安装 2](#_Toc21922)

[第2次实验 Linux常用命令 4](#_Toc8553)

[第3次实验 Linux系统C程序的编写 10](#_Toc31028)

[第4次实验 Linux进程管理 12](#_Toc21856)

[第5次实验 进程调度算法模拟 14](#_Toc5985)

[第6次实验 银行家算法 16](#_Toc2625)

# 第1次实验 Linux系统安装

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **Linux系统安装** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2022210901 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的   1、了解Linux系统；  2、掌握虚拟机软件VMWare WorkStation的安装方法（MAC系统需安装（Virtual Box）；  3、掌握利用VMWare虚拟机安装Linux的方法。 | | | |
| 二、实验内容  1、VMWare WorkStation的安装  2、VMWare下新建虚拟机  3、利用VMWare虚拟机安装Linux  4、安装VMWare Tools | | | |
| 三、实验结果截图  1、VMWare软件界面截图    2、安装Linux重要步骤截图    IMG_256 | | | |

# 第2次实验 Linux常用命令

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **Linux常用命令使用（1）** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/9/25 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的 2. 了解Linux命令的普遍语法格式； 3. 了解Linux目录常见概念；   2、熟练掌握常用命令和选项的功能。 | | | |
| 二、实验内容  练习常用的Linux Shell 命令及命令选项，要求熟练掌握下列命令的使用。  （1）显示及改变目录命令：pwd、cd、ls  （2）文件及目录的创建、复制、删除和移动命令：touch、mkdir、rmdir、cp、mv、rm  练习题目：   1. 使用命令切换到/etc目录，并显示当前工作目录路径。 2. 使用命令显示/root目录下所有文件和子目录的详细信息。 3. 使用命令创建空文件/root/ab,然后将/etc/named.conf文件复制到该目录中，最后再将该目录及其目录下的文件一起删除。 | | | |
| 三、实验结果截图 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **Linux常用命令使用（2）** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/10/9 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的 2. 掌握文件处理命令；   2、掌握文件查找命令。 | | | |
| 二、实验内容  练习常用的Linux Shell 命令及命令选项，要求熟练掌握下列命令的使用。   1. 文件处理命令：touch、cat、more、less、head、tail 2. 文件查找命令：find   练习题目：   1. 使用命令一次三行显示/etc/named.conf文件的内容。 2. 使用cat命令创建mm.txt文件，文件内容为Hello。 3. 使用命令查找/etc目录下的文件named.conf。 4. 熟悉使用find中的通配符查找相关文件。 | | | |
| 三、实验结果截图 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **Linux常用命令使用（3）** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/12/4 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的 2. 掌握文件/目录访问权限管理命令；   2、掌握文件/目录的打包和压缩命令。 | | | |
| 二、实验内容  练习常用的Linux Shell 命令及命令选项，要求熟练掌握下列命令的使用。   1. 文件/目录访问权限管理命令：chmod、umask、chown 2. 文件/目录的打包和压缩命令：gzip、tar、zip、unzip   练习题目：   1. 使用命令创建具有默认权限为744的目录/root/ak。 2. 将当前用户主目录打包成tar.zg格式备份，并将该文件权限设置为666。 3. 在当前目录下创建backup目录，并将上题中的tar.gz文件解压缩到该目录。 | | | |
| 三、实验结果截图 | | | |

# 第3次实验 Linux系统C程序的编写

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **Linux下C程序的编写** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/10/23 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的   1、掌握Linux 下C 程序的编写、编译与运行方法。  2、掌握gcc 编译器的编译过程，熟悉编译的各个阶段。  3、熟悉Makefile 文件的编写格式和make 编译工具的使用方法。 | | | |
| 二、实验内容  练习使用gcc 编译器编译C 程序并执行，编写Makefile 文件，使用make 工具编译程序并执行。  具体要求：  （1）编写简单的C 程序，功能为在屏幕上输出“Hello gcc！”。利用该程序练习使用gcc 编译器的E、S、c、o、g 选项，观察不同阶段所生成的文件，即\*.c、\*.i、\*.s、\*.o 文件和可执行文件。  （2）编写一个由头文件greeting.h、自定义函数文件greeting.c、主函数文件myapp.c 构成的C 程序，并根据这三个文件的依赖关系编写Makefile 文件。 | | | |
| 三、实验结果截图 | | | |

# 第4次实验 Linux进程管理

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **进程控制** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/12/4 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的 2. 加深对进程概念的理解，进一步认识并发执行的实质。 3. 掌握Linux 操作系统中进程的创建和终止操作。 4. 掌握在Linux 操作系统中创建子进程并加载新映像的操作。 | | | |
| 二、实验内容  （1）编写一个C 程序，并使用系统调用fork()创建一个子进程。要求如下：  ①在子进程中分别输出当前进程为子进程的提示、当前进程的PID 和父进程的PID、根据用户输入确定当前进程的返回值、退出提示等信息。  ②在父进程中分别输出当前进程为父进程的提示、当前进程的PID 和子进程的PID、等待子进程退出后获得的返回值、退出提示等信息。  （2）编写另一个C 程序，使用系统调用fork()以创建一个子进程，并使用这个子进程调用exec 函数族以执行系统命令ls。 | | | |
| 三、实验结果截图 | | | |

# 第5次实验 进程调度算法模拟

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **进程调度算法模拟实验** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/11/6 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 1. 实验目的   （1）加深对进程概念的理解，明确进程和程序的区别；  （2）深入理解系统如何组织进程；  （3）理解常用进程调度算法的具体实现。 | | | |
| 二、实验内容  编写C程序模拟实现单处理机系统中的进程调度算法，实现对多个进程的调度模拟，要求采用常见进程调度算法（如先来先服务、时间片轮转和优先级调度等算法）进行模拟调度。  思考：如何在基于动态优先数的进程调度算法来实现先来先服务和时间片轮转调度算法？ | | | |
| 三、实验结果截图 | | | |

# 第6次实验 银行家算法

|  |  |  |  |
| --- | --- | --- | --- |
| **实验任务** | **避免死锁（银行家算法）** | | |
| **姓名** | 唐诗益 | **操作性质** | 基本操作 （ ） 验证性操作（ ）  综合性操作（ ） 设计性操作（ ） |
| **日期** | 2024/12/4 | **成绩** |  |
| 教师评价：  实验预习（ ） 实验操作（ ）实验结果（ ）实验报告（ ）  教师签名： | | | |
| 一、实验目的   1. 加深了解有关资源申请，避免死锁等概念。 2. 体会和了解死锁和避免死锁的具体实施方法。 3. 了解死锁的相关知识。 4. 掌握银行家算法、系统安全性检查算法。 | | | |
| 二、实验内容  银行家算法（Banker’s Algorithm）是一个避免进程死锁的著名算法。通过银行家算法，我们可以有效地避免死锁发生的情况。请根据银行家算法完成下列题目。   1. 设系统中有3种类型的资源（A,B,C）和5个进程P1、P2、P3、P4、P5，其中Available=（2,3,3）。在T0时刻系统状态见下表（T0时刻系统状态表）所示。   T0时刻系统状态表   |  |  |  | | --- | --- | --- | |  | 资源需求量 | 已分配资源数量 | | A B C | A B C | | P1 | 3 4 7 | 2 1 2 | | P2 | 1 3 4 | 4 0 2 | | P3 | 0 0 3 | 3 0 5 | | P4 | 2 2 1 | 2 0 4 | | P5 | 1 1 0 | 3 1 4 |   请求序列如下： a：进程P2请求资源（0，3，4） b：进程P4请求资源（1，0，1） c：进程P1请求资源（2，0，1） d：进程P3请求资源（0，0，2）  试问：系统能否将进程请求的资源分配给它？  任务1：认真阅读程序，将程序执行的流程描述出来。  任务2：修改程序，要求如果资源分配成功，将安全序列输出。 | | | |
| 三、主要操作代码整理与实验结果展示  （一）源程序（实验步骤/实验过程/算法）  #include<iostream>  #include<string>  using namespace std;  #define n 5  #define m 3  int available[m] = { 2,3,3 }, alloc[n][m], need[n][m], request[n][m];  void init()  {  alloc[0][0] = 2; alloc[0][1] = 1; alloc[0][2] = 2;  alloc[1][0] = 4; alloc[1][1] = 0; alloc[1][2] = 2;  alloc[2][0] = 3; alloc[2][1] = 0; alloc[2][2] = 5;  alloc[3][0] = 2; alloc[3][1] = 0; alloc[3][2] = 4;  alloc[4][0] = 3; alloc[4][1] = 1; alloc[4][2] = 4;    need[0][0] = 3; need[0][1] = 4; need[0][2] = 7;  need[1][0] = 1; need[1][1] = 3; need[1][2] = 4;  need[2][0] = 0; need[2][1] = 0; need[2][2] = 3;  need[3][0] = 2; need[3][1] = 2; need[3][2] = 1;  need[4][0] = 1; need[4][1] = 1; need[4][2] = 0;    request[0][0] = 2; request[0][1] = 0; request[0][2] = 1;  request[1][0] = 0; request[1][1] = 3; request[1][2] = 4;  request[2][0] = 0; request[2][1] = 0; request[2][2] = 2;  request[3][0] = 1; request[3][1] = 0; request[3][2] = 1;    }  void display()  { int i,j,k;  cout << "资源 可分配" << endl;  for ( i = 0; i < m; i++)  {  cout << (char)('A' + i) << " " << available[i] << endl;  }  cout << endl;  cout << "进程 资源 已分配 需求"<<endl;  for (j = 0; j < n; j++)  {    for (k = 0; k < m; k++)  {  cout << j+1<< " " <<(char)('A' + k)<< " " << alloc[j][k] << " " << need[j][k] << endl;  }  }  cout << endl;  }  bool judge()  {  int i, j, tag = n;  bool non;  int work[m], finish[n];  for (i = 0; i<m; i++) work[i] = available[i];  for (i = 0; i<n; i++) finish[i] = 0;  while (tag--)  {  for (i = 0; i<n; i++)  {  if (finish[i] == 0)  {  non = true;  for (j = 0; j < m; j++)  {  if (need[i][j] > work[j]) non = false;  }  if(non)  {  for (j = 0; j < m; j++)  {  work[j] = work[j] + alloc[i][j];  finish[i] = 1;  }  }  }  }  }  non = true;  for (i = 0; i < n; i++)  {  if (finish[i] != 1) non = false;  }  if (non) return 1;  else return 0;  }  void deal(int k)  {  for (int j = 0; j < m; j++)  {  if (request[k][j] > need[k][j])  {    cout << " 非法！！！" << endl;  return;  }  }  for (int w = 0; w< m; w++)  {  if (request[k][w] > available[w])  {    cout << " 阻塞！！！" << endl;  return;  }  }  for (int p = 0; p < m; p++) //试探性分配  {  available[p] -= request[k][p];  alloc[k][p] += request[k][p];  need[k][p] -= request[k][p];  }  if (judge())  {    cout << " 资源分配成功！！！" << endl << endl;  display();  }  else  {    cout << " 资源分配后系统处于不安全状态！！！" << endl;  for (int j = 0; j < m; j++) //试探性分配还原  {  available[j] += request[k][j];  alloc[k][j] -= request[k][j];  need[k][j] += request[k][j];  }  }  cout << endl;  }  int main()  {  init();  display();  cout << "a: 进程p2请求资源（0，3，4） " ;  deal(1);  cout << "b: 进程p4请求资源（1，0，1） " ;  deal(3);  cout << "c: 进程p1请求资源（2，0，1） " ;  deal(0);  cout << "d: 进程p3请求资源（0，0，2） " ;  deal(2);  return 0;  }  （二）主要操作步骤截图展示 | | | |