|  |
| --- |
| 北 京 邮 电 大 学  实 验 报 告  课程名称： Linux 环境及开发工具应用实践  院系：计算机学院（国家示范性软件工程学院）  班级： 2021211318  姓名： 万志恒  学号： 2021212413  教师： 赵方 金昕  成绩：  2023年 春季学期 |
| 实验目的 （1）理解进程的概念，明确进程和程序的区别。  （2）理解并发执行的实质。  （3）掌握进程的睡眠、同步、撤销等进程控制方法。 实验环境 硬件：PC微型计算机、8核、16.0G内存、467G硬盘  软件：Vmware、Ubuntu操作系统   1. **实验任务及内容**   2.1. 进程的创建。  ① 编写一段源程序，使系统调用fork()创建两个子进程，当此程序运行时，在系统中 有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符：父进程显示字符 “a”；子进程分别显示字符“b”和字符“c”。试观察记录屏幕上的显示结果，并分析原因。  源程序：  #include <sys/types.h>  #include <stdio.h>  #include <unistd.h>  #include <sys/wait.h>  int main(){  pid\_t pid;  pid = fork();  if(pid<0){//error  fprintf(stderr, "Fork Failed");  return 1;  }else if(pid==0){//child process  printf("b");  }else{//parent process  pid = fork();  if(pid<0){  fprintf(stderr, "Fork Failed");  return 1;  }else if(pid==0){//another child process  printf("c");  }else{  printf("a");  wait(NULL);  }  }  return 0;  }  显示结果：    分析原因：一旦启动子进程，后续的代码就并发，没有先后顺序，具体取决于调度策略和进程的优先级。  ② 修改已编写的程序，将每个进程输出一个字符改为每个进程输出一句话，在观察程序执行时屏幕出现的现象，并分析原因。  源程序：  #include <sys/types.h>  #include <stdio.h>  #include <unistd.h>  #include <sys/wait.h>  int main(){  pid\_t pid;  pid = fork();  if(pid<0){//error  fprintf(stderr, "Fork Failed");  return 1;  }else if(pid==0){//child process  printf("I’m b.");  }else{//parent process  pid = fork();  if(pid<0){  fprintf(stderr, "Fork Failed");  return 1;  }else if(pid==0){//another child process  printf("I’m c.");  }else{  printf("I’m a.");  wait(NULL);  }  }  return 0;  }  显示结果：    分析原因：printf()函数会将待输入的内容一并放入缓冲区，随后一次性输出，因而不会出现不同句子的字符交叉出现的情况。  2.2. 进程的控制 ① 用fork()创建一个进程，再调用exec()用新的程序替换该子进程的内容。 ② 利用wait()来控制进程执行顺序。  源程序：  #include <sys/wait.h>  #include <stdlib.h>  #include <sys/types.h>  #include <stdio.h>  #include <unistd.h>  int main(){  pid\_t pid;  pid=fork();  if(pid<0){  fprintf(stderr, "Fork Failed");  return 1;  }else if(pid==0){  execl("/bin/ls","ls","-1","-color", NULL);  printf("exec fail!/n");  exit(1);  }else{  wait(NULL);  printf("is completed!/n");  exit(0);  }  }  显示结果：    分析原因：父进程创建子进程后，由于wait(NULL)的存在，其会等待子进程执行完毕后再执行。子进程中execl()函数将子进程的内容替换为参数对应的信息，随后执行。当子进程执行完毕后，父进程继续执行输出“is competed”。 实验心得及体会 通过这两个实验，我加深了对父进程与子进程、fork()函数等内容的了解，同时对创建子进程以及父进程和子进程的执行过程有了更加直观的认识。 |