|  |
| --- |
| 北 京 邮 电 大 学  实 验 报 告  课程名称： 操作系统原理  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  成绩：  2023年 春季学期 |
| 实验目的 1. 理解 Linux 管理进程所用到的数据结构。  2. 理解 Linux 的进程调度算法的处理逻辑及其实现所用到的数据结构。 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、Vi、GCC、GDB   1. **实验任务及内容**   1. 通过查阅参考书或者上网找资料，熟悉/usr/src/linux（注意：这里最后一级目录名可能是个含有具体内核版本号和“linux”字符串的名字）下各子目录的内容，即所含 Linux源代码的情况。  2. 分析 Linux 进程调度有关函数的源代码，主要是 schedule()函数，并且要对它们引用的头文件等一并分析。  3. 实现 Linux 的进程调度算法及理解其实现所用的主要数据结构。  （可参考Linux0.11版本：https://github.com/Wangzhike/HIT-Linux-0.11  或者<https://github.com/tinyclub>）  **本人负责任务三的FIFO、Priority调度算法。**  **任务三：**  **FIFO调度算法**  **源程序：**  #include<stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<unistd.h>  #include<pthread.h>  #include<time.h>  #include<iostream>  #define Thread\_Num 20  **using** **namespace** std**;**  pthread\_mutex\_t Device\_mutex**;**  **struct** VirtualPCB  **{**  **int** tid**;**  **int** priority**;**  **int** waittime**;**  **int** runtime**;**  **int** arrivetime**;**  **int** visited**;**  **int** tempruntime**;**  **public:**  **int** gettid**()**  **{**  **return** tid**;**  **}**  **int** getwaittime**()**  **{**  **return** waittime**;**  **}**  **int** getpriority**()**  **{**  **return** priority**;**  **}**  **int** getruntime**()**  **{**  **return** runtime**;**  **}**  **int** getarrivetime**()**  **{**  **return** arrivetime**;**  **}**  **void** setvisit**(int** a**)**  **{**  visited**=**a**;**  **}**  **int** getvisit**()**  **{**  **return** visited**;**  **}**  **int** gettempruntime**()**  **{**  **return** tempruntime**;**  **}**  **void** setwaittime**(int** n**)**  **{**  waittime **=** n**;**  **}**  **void** settempruntime**(int** n**)**  **{**  tempruntime **=** tempruntime **-** n**;**  **}**  **}**TCB**[**Thread\_Num**];**  *//PCB初始化函数*  **void** t\_init**()**  **{**  **int** n**;**  srand**(**time**(**NULL**));**  **for(**n **=**0**;**n**<**Thread\_Num**;**n**++)**  **{**  TCB**[**n**].**tid **=** n **+** 1**;***//用线程创建序号作为虚拟id*  *//用随机数随机产生虚拟PCB的值*  TCB**[**n**].**priority **=** 1 **+** rand**()%**19**;**  TCB**[**n**].**runtime **=** 1 **+** rand**()%**19**;**  TCB**[**n**].**arrivetime **=** rand**()%**201**;***//假定所有线程在0~500时间内相继就绪*  TCB**[**n**].**waittime **=** 0**;**  TCB**[**n**].**visited **=**0**;**  TCB**[**n**].**tempruntime **=** TCB**[**n**].**runtime**;**  **}**  **}**  *//线程运行内容*  **void** **\***t\_print**(void** **\***arg**)**  **{**  **int** n **=** **\*(int** **\*)**arg**;***//获取当前线程id*  **while(**1**)**  **{**  pthread\_mutex\_lock**(&**Device\_mutex**);**  printf**("Thread\_%-2d: ",** n**);**  printf**("tid:%-2d priority:%-2d arrivetime:%-2d runtime:%-2d \n",**TCB**[**n**-**1**].**gettid**(),** TCB**[**n**-**1**].**priority**,** TCB**[**n**-**1**].**arrivetime**,** TCB**[**n**-**1**].**runtime**);**  pthread\_mutex\_unlock**(&**Device\_mutex**);**  sleep**(**1**);**  **break;**  **}**  pthread\_exit**(**0**);**  **}**  *//FIFO*  **void** FIFO**()**  **{**  cout**<<"-----------FIFO:"<<**endl**;**  **int** i**,**j**;**  **int** start **=** 0**;**  **float** waittime **=** 0**;**  **float** avwait **=** 0**;**  **for(**i**=**0**;**i**<**201**;**i**++)**  **{**  **for(**j**=**0**;**j**<**Thread\_Num**;**j**++){**  **if(**TCB**[**j**].**getarrivetime**()==**i **&&** TCB**[**j**].**getvisit**()==**0**){**  **if(**start**<**TCB**[**j**].**getarrivetime**())** start **=** TCB**[**j**].**getarrivetime**();***//空转情况*  printf**("Thread: %-2d Arrive: %-3d Start: %-3d Runtime: %-2d\n",**  TCB**[**j**].**gettid**(),**TCB**[**j**].**getarrivetime**(),**start**,**TCB**[**j**].**getruntime**());**  TCB**[**j**].**waittime**=(float)**start **-** TCB**[**j**].**getarrivetime**();**  waittime**+=**TCB**[**j**].**getwaittime**();**  start **=** start **+** TCB**[**j**].**getruntime**();**  TCB**[**j**].**setvisit**(**1**);**  **}**  **}**  **}**  avwait **=** waittime **/** **(float)**Thread\_Num**;**  printf**("Total waitting time : %f\n",**waittime**);**  printf**("Average waitting time : %f\n",**avwait**);**  **}**  *//创建线程函数*  **void** **\***Children**(void\*)**  **{**  **int** ret**[**Thread\_Num**];**  t\_init**();**  pthread\_t tid**[**Thread\_Num**];**  pthread\_mutex\_init**(&**Device\_mutex**,**NULL**);**  **int** i**,**j**;**  **for(**i**=**0**;**i**<**Thread\_Num**;**i**++)**  **{**  **int** k **=**i**+**1**;**  ret**[**i**]** **=** pthread\_create**(&**tid**[**i**],**NULL**,&**t\_print**,** **&**k**);**  **if(**ret**[**i**]** **==** 0**)** **{**  sleep**(**1**);**  **}**  **else{**  printf**("Thread\_%-2d failed!\n",**i**+**1**);**  **}**  **}**  **for(**j**=**0**;**j**<**Thread\_Num**;**j**++)**  pthread\_join **(**tid**[**i**],** NULL**);**  pthread\_mutex\_destroy**(&**Device\_mutex**);**  pthread\_exit**(**0**);**  **}**  **int** main**()**  **{**  **int** ret1**;**  pthread\_t tid1**;***//主线程*  ret1 **=** pthread\_create**(&**tid1**,**NULL**,&**Children**,**NULL**);**    **if(**ret1 **==** 0**)**  **{**  printf**("Main Thread done\n");**  sleep**(**20**);**  **}**  **else{**  printf**("Thread failed!\n");**  **}**  FIFO**();**  **return** 0**;**  **}**  **运行结果：**      **Priority调度算法：**  **源程序：**  #include<stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<unistd.h>  #include<pthread.h>  #include<time.h>  #include<iostream>  #define Thread\_Num 20  **using** **namespace** std**;**  pthread\_mutex\_t Device\_mutex**;**  **struct** VirtualPCB  **{**  **int** tid**;**  **int** priority**;**  **int** waittime**;**  **int** runtime**;**  **int** arrivetime**;**  **int** visited**;**  **int** tempruntime**;**  **public:**  **int** gettid**()**  **{**  **return** tid**;**  **}**  **int** getwaittime**()**  **{**  **return** waittime**;**  **}**  **int** getpriority**()**  **{**  **return** priority**;**  **}**  **int** getruntime**()**  **{**  **return** runtime**;**  **}**  **int** getarrivetime**()**  **{**  **return** arrivetime**;**  **}**  **void** setvisit**(int** a**)**  **{**  visited**=**a**;**  **}**  **int** getvisit**()**  **{**  **return** visited**;**  **}**  **int** gettempruntime**()**  **{**  **return** tempruntime**;**  **}**  **void** setwaittime**(int** n**)**  **{**  waittime **=** n**;**  **}**  **void** settempruntime**(int** n**)**  **{**  tempruntime **=** tempruntime **-** n**;**  **}**  **}**TCB**[**Thread\_Num**];**  *//PCB初始化函数*  **void** t\_init**()**  **{**  **int** n**;**  srand**(**time**(**NULL**));**  **for(**n **=**0**;**n**<**Thread\_Num**;**n**++)**  **{**  TCB**[**n**].**tid **=** n **+** 1**;***//用线程创建序号作为虚拟id*  *//用随机数随机产生虚拟PCB的值*  TCB**[**n**].**priority **=** 1 **+** rand**()%**19**;**  TCB**[**n**].**runtime **=** 1 **+** rand**()%**19**;**  TCB**[**n**].**arrivetime **=** 0**;***//假定所有线程在0时刻依次就绪*  TCB**[**n**].**waittime **=** 0**;**  TCB**[**n**].**visited **=**0**;**  TCB**[**n**].**tempruntime **=** TCB**[**n**].**runtime**;**  **}**  **}**  *//线程运行内容*  **void** **\***t\_print**(void** **\***arg**)**  **{**  **int** n **=** **\*(int** **\*)**arg**;***//获取当前线程id*  **while(**1**)**  **{**  pthread\_mutex\_lock**(&**Device\_mutex**);**  printf**("Thread\_%-2d: ",** n**);**  printf**("tid:%-2d priority:%-2d runtime:%-2d \n",**TCB**[**n**-**1**].**gettid**(),** TCB**[**n**-**1**].**priority**,** TCB**[**n**-**1**].**arrivetime**,** TCB**[**n**-**1**].**runtime**);**  pthread\_mutex\_unlock**(&**Device\_mutex**);**  sleep**(**1**);**  **break;**  **}**  pthread\_exit**(**0**);**  **}**  *//优先级调度*  **void** Priority**()**  **{**  cout**<<"-----------Priority:"<<**endl**;**  **int** i**,**j**;**  **int** start **=** 0**;**  **float** waittime **=** 0**;**  **float** avwait **=** 0**;**  **for(**i**=**1**;**i**<**Thread\_Num**;**i**++)**  **{**  **for(**j**=**0**;**j**<**Thread\_Num**;**j**++){**  **if(**TCB**[**j**].**getpriority**()==**i **&&** TCB**[**j**].**getvisit**()==**0**){**  printf**("Thread: %-2d Priority: %-2d Start: %-3d Runtime: %-2d\n",**  TCB**[**j**].**gettid**(),**TCB**[**j**].**priority**,** start**,** TCB**[**j**].**getruntime**());**  waittime **=** waittime **+** **(float)**start**;**  start **=** start **+** TCB**[**j**].**getruntime**();**  TCB**[**j**].**setvisit**(**1**);**  **}**  **}**  **}**  avwait **=** waittime **/** **(float)**Thread\_Num**;**  printf**("Total waitting time : %f\n",**waittime**);**  printf**("Average waitting time : %f\n",**avwait**);**  **}**  *//创建线程函数*  **void** **\***Children**(void\*)**  **{**  **int** ret**[**Thread\_Num**];**  t\_init**();**  pthread\_t tid**[**Thread\_Num**];**  pthread\_mutex\_init**(&**Device\_mutex**,**NULL**);**  **int** i**,**j**;**  **for(**i**=**0**;**i**<**Thread\_Num**;**i**++)**  **{**  **int** k **=**i**+**1**;**  ret**[**i**]** **=** pthread\_create**(&**tid**[**i**],**NULL**,&**t\_print**,** **&**k**);**  **if(**ret**[**i**]** **==** 0**)** **{**  sleep**(**1**);**  **}**  **else{**  printf**("Thread\_%-2d failed!\n",**i**+**1**);**  **}**  **}**  **for(**j**=**0**;**j**<**Thread\_Num**;**j**++)**  pthread\_join **(**tid**[**i**],** NULL**);**  pthread\_mutex\_destroy**(&**Device\_mutex**);**  pthread\_exit**(**0**);**  **}**  **int** main**()**  **{**  **int** ret1**;**  pthread\_t tid1**;***//主线程*  ret1 **=** pthread\_create**(&**tid1**,**NULL**,&**Children**,**NULL**);**    **if(**ret1 **==** 0**)**  **{**  printf**("Main Thread done\n");**  sleep**(**20**);**  **}**  **else{**  printf**("Thread failed!\n");**  **}**  Priority**();**  **return** 0**;**  **}**  **实验结果：**     实验心得及体会 学习了Linux操作系统中的三种调度，通过阅读源码，更加深刻地认识了这三种调度地实现方式。  手动实现FIFO算法和Priority算法，锻炼了我的代码能力，同时加深了我对FIFO算法和Priority算法的印象以及对他们各自特点的认识。 |