Development of Multitask Application on Windows

Fan Ce, Li Chang
Faculty of Information Science and Technology, Guangdong University of Foreign Studies, GuangZhou, China
fancemail@yahoo.com.cn

Abstract: Operating systems are not only abstract on theory but also complicated on implementation. A development schemata is shown based on the multitask application in this paper. It can be considered as a model for the development of time-sharing, or real-time application. Also concurrent execution of multitask is an important feature in any operating system. Thus, in order to meet the needs of the development of applying system from time-sharing or real-time multitask under Windows, this paper shows the development schemata of concurrent execution of multitask by C programming in the Windows operating system. Also, this paper presents A method of concurrent execution of multitask using clock interruption based on Windows. The schemata realizes simultaneous execution of multitask in a system , which works out the general method and structure of multitask of time-sharing or real-time based on Windows for user-application system.

Key word: multitask; concurrency; interruption; clock

1 Introduction

Nowadays operating systems are not only very complexity but also powerful. Operating systems of Windows series become the development platform of application systems and teaching, which help us to develop and design a schemata of multitasks application systems based on the Windows, and understand the change of multi activity and states in a system using interruption mechanism. This paper shows a basic method developing for multitask programming on Windows.

The principle of scheduling mechanism for Windows and its development schemata are shown in section 2. A method of multitask programming using time-sharing and some codes are shown in section 3. The conclusion is presented in section 4.

2 Scheduling Mechanism and Multitask Application on Windows

Because there exists the feature of time-sharing in modern operating systems, clock is essential in a system. Fig.1(a) shows all kinds of interruption and its priorities on Windows(2000/xp). Fig.1(b) describes the preemption of a multitask application to the clock-timing of the system, so that it makes a timing interruption to trigger system timing indirectly. The basic characteristics of scheduling mechanism on Windows follow as:

- The scheduling unit of Windows 2000/xp

Fig.1. Queue and timing clock based on priority (a) Windows interrupt and priority (b) the application schemata of multitask.
processors is thread instead of processes. Strategies of scheduling-threads are preemptive based on priority and time-slice given, and scheduling by RR(Round-Robin) at the same level.

- When a thread is running, its running time is a time-slice called as a quantum. The quantum is not time-length but a only unit-integer. Once a clock interruption occurs, the value is decreased by a fixed number.

It, from the Fig.1, can be seen that timing interruptions of a system are used as foundation of CPU scheduling by Windows. So, deprivation of a system timing may make running of other processes in Windows unstable.

While using clock interruptions in it as tasks switch among a multitask application after understanding the scheduling mechanism, we are sure that original scheduling of the system is ordinary. What we do is that a multitask application is to be considered as a applying process in Windows for that(see Fig.1(b)schemata ). Thus, the timing interruption is used as a trigger to the multitask application directly.

3 Implementation of Multitask Time-Sharing

Not only running of many tasks in systems have themselves space of codes and data from system, but also it is important to have the protection of current work set of CPU and implementation of scheduling algorithms on thread schemata of user-level.

This design model establishes a concurrent execution of multitask by using the system clock.

3.1 Implementation of Multitask

The aim of this multitask application mainly describes a method of implementation of concurrent multitask, while what tasks are not important. There may be many tasks running in turn controlled by the RR scheduling algorithm of a application system by using system clock shown in this model.

The method of multitask application creates 4 tasks(or called as terminal user) using C programming in a Windows, where each rectangle indicates a task ,or terminal user. From the Fig.2, 4 tasks are:

- A task of displaying a random character at random position and color in the window of upper-left corner.
- A task of displaying a rectangle at random two corner points and color in the window of upper-right corner.
- A task of displaying the “Hello! Window is running” at random color in the window of bottom-left corner.
A task of receive characteritics from a keyboard online in the window of bottom-right corner.

The Fig.3 shows a schemata of time-sharing operating of 4 tasks in Windows. These 4 tasks are running simultaneously as whole, but only as a process in Windows. Thus, they are transparent to the Windows system.

Main functions and data structure of the method creating many tasks by C programming follow as:

```c
make_task(taskptrtask,unsigned stck,unsigned id) /* create a task in memory */
{struct int_regs *r;
  if(id>=NUM_TASKS||id<0) return 0;
  disable();
  tasks[id].stck=malloc(stck+sizeof(struct int_regs));
  r = (struct int_regs *) tasks[id].stck+stck-sizeof(struct int_regs);
  tasks[id].sp = FP_OFF((struct int_regs far *) r);
  tasks[id].ss = FP_SEG((struct int_regs far *) r);
  r->cs = FP_SEG(task);
  r->ip = FP_OFF(task);
  r->ds =_DS; r->es =_ES;
  tasks[id].status = READY;
  enable();
  return 1;
}
```

Where disable() and enable() are the open and close of system interrupts respectively, and r is used to build the stack segment and starting location of program for each task, which ensure the independence and protection and restoration of CPU work set for each task. Fig.4 shows a ordinal model. The multitask design developed in this paper responding to user-level threads shown in Fig.4(b).

### 3.2 Interrupt and Scheduling

As main aim of the schemata realizes switch among tasks in a applying process using interrupts of the system clock, it only refers to CPU execution turn among tasks so as to result in “simultaneous working”. Thus, programming based on the method will consider and implement the switch and scheduling algorithms for multitask in a applying system. It can be completed with the help of system interrupts and scheduling algorithms designed by programmers. The schemata is helpful for demanding to create a applying system with many tasks in Windows systems.

Main functions for interrupt processing and scheduling follow as:

```c
void interrupt multitask(void) /* establish own entry of interrupt processing */
{ disable();
  old_int8 =getvect(8); setvect(8,int8_task_switch);
  oldss = _SS; oldsp = _SP;
  _SS = tasks[tswitch].ss;
  SP = tasks[tswitch].sp;
  enable();
}
void interrupt int8_task_switch(void) /* the
interrupt processing and scheduling by RR */
{ (*old_int8());
  if(single_task) return;
tasks[tswitch].ss = _SS;
tasks[tswitch].sp = _SP;
save current CPU set for a task interrupted;
search a task in a ready queue;
---
_SS = tasks[tswitch].ss;
_SP = tasks[tswitch].sp;
tasks[tswitch].status = RUNNING;
}

where interrupt indicates that the function is a function
for interrupt processing, and int8_task_switch is used as
switch among multitask scheduling.

4 Conclusion

Operating systems are not only abstract on theory
but also complicated on implementation. A development
schemata is shown based on the multitask application in
this paper. It can be considered as a model for the
development of time-sharing, or real-time application,
and achieve two objectives:

- A applying system can be implemented if only each
task is changed, so it can be as a “template” of
multitask application in Windows.
- Also, it can be a experiment illustration for anybody
studying concurrent working of many processes in a
operating system and designing own scheduling
algorithms from this method. People not only
understand the principle of system concurrency but
also know well the development of applying
systems of multitask in Windows.

It should be emphasized on a difference between
the schemata described in this paper and system calls
used to implement concurrent multitask. The former is
able to realize the scheduling among tasks/threads and is
a system problem, the latter is how to use system calls to
create many processes which are scheduled by operating
systems after these commands or functions are studied
seriously by programmers.

Acknowledgment

This work is supported by the Natural Science
Foundation of Guangdong University of Foreign Studies
under Grant No. GW2006-TB-012

References

principles[M].(Fifth edition). Translated by Chen Yu. Peking:
[2] Fan Ce. A Course in computer operating systems-internals and
[7] Zhang Xuerao. A Course in computer operating systems. Peking: