Slide 1

Threads and Processes — Part 1

COMP3231/COMP9201 Operating Systems

2003/S2

Slide 2

Major Requirements of an OS

- ➜ Interleave the execution of several programs
  - to maximize utilization of CPU and other resources while providing reasonable response time
  - to support multiple user working interactively
  - for convenience (e.g., compile program while editing other file)
- ➜ Allocate resources required for execution of programs
- ➜ Support communication between executing programs

Slide 3

Previously, we listed several definitions of the term Process:

- ✶ A program in execution
- ✶ An instance of a program running on a computer
- ✶ A unit of execution characterised by
  - a single, sequential thread of execution
  - a current state
  - an associated set of system resources (memory, devices, files)
- ✶ Unit of resource ownership

Many applications consist of more than one thread of execution which share resources
⟹ distinction between thread and process

Slide 4

Processes and Threads

Process:

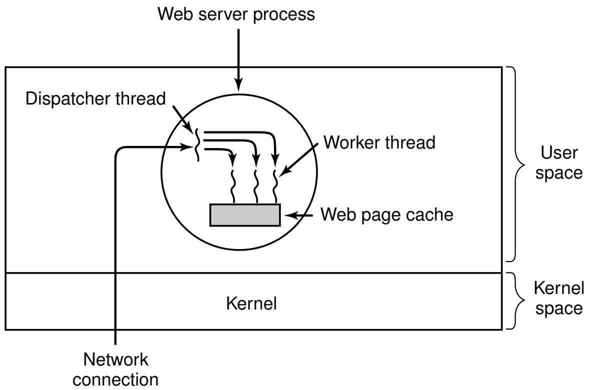
- ➜ "Owner" of resources allocated for individual program execution
- ➜ Can encompass more than one thread of execution
  - Outlook, Evolution: different threads for calendar, mail components etc

Thread:

- ➜ Unit of execution
- ➜ Belongs to a process
- ➜ Can be traced
  - list the sequence of instructions that execute

Slide 5

EXAMPLE: WEB SERVER



Slide 6

SINGLE-THREADED WEB SERVER IMPLEMENTATIONS

➜ Sequential processing of requests:
- web server gets request, processes it, accepts next request
- CPU idle while data retrieved from disk
- Poor performance

➜ Finite-State Machine:
- use non-blocking read
- program records state of current request
- gets next event
- on reply (signal) from disk, fetches and processes data
- good performance, complicated to implement and debug

➜ Processes instead of Threads
- Communicate by sharing data, messages

Slide 7

ADVANTAGES OF THREADS

① Program does not stall when one of its operations blocks
- save contents of a page to disk while downloading other page

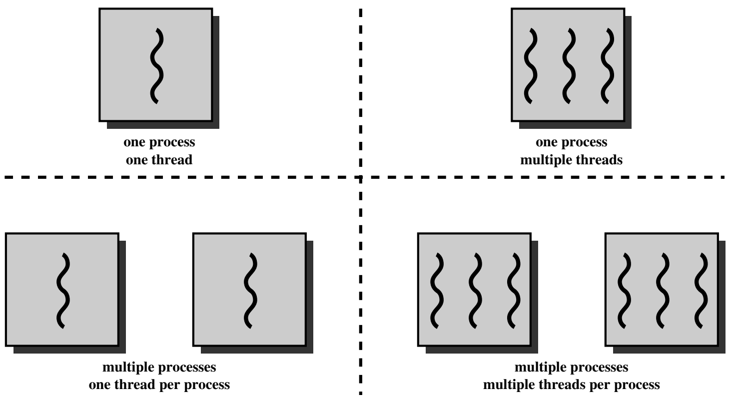
② Overhead for thread creation and destruction is less than for processes (depending on implementation, can be about a factor of 100 faster)

③ Simplification of programming model

④ Performace gains on machines with multiple CPU's

Slide 8

THREADS AND PROCESSES



Slide 9

Threads and Processes

- ➜ Single process, single thread
  - MS-DOS, old MacOS
- ➜ Single process, multiple threads
  - OS/161 as distributed
- ➜ Multiple processes, single thread
  - traditional Unix
- ➜ Multiple processes, multiple threads
  - modern Unices (Solaris, Linux), Windows-2000

Note: Literature (incl. textbooks) often do not cleanly distinguish those concepts (for historical reasons)!

Slide 10

Logical traces of threads:

Thread A	Thread B	Thread C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000: Starting address of code for Thread A

8000: Starting address of code for Thread B

12000:Starting address of code for Thread C

Slide 11

Logical traces of threads:

1	5000	
2	5001	
3	5002	
4	5003	
5	5004	
6	5005	
		Time out
7	100	
8	101	
9	102	
10	103	
11	104	
12	105	
13	8000	
14	8001	
15	8002	
16	8003	
17	8004	
		I/O request

18	100	
19	101	
20	102	
21	103	
22	104	
23	105	
24	12000	
25	12001	
26	12002	
27	12003	
28	12004	
29	12005	
		Time out
30	100	
31	101	
32	102	
33	103	
34	104	
35	105	

Slide 12

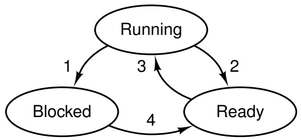
Thread States

Three states (may be more, depending on implementation):

① Running: currently active, using CPU

② Ready: runnable, waiting to be scheduled

③ Blocked: waiting for an event to occur (I/O, alarm)



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Slide 13

Reasons for Leaving the Running State

- Thread terminates
  - exit() system call (voluntary termination)
  - killed by another thread
  - killed by OS (due to exception)
- Thread cannot continue execution
  - blocked waiting for event (I/O)
- OS decides to give someone else a chance
  - requires the OS to be invoked
    - via system call or exception
    - via interrupt
- Thread voluntarily gives another thread a chance
  - yield() system call

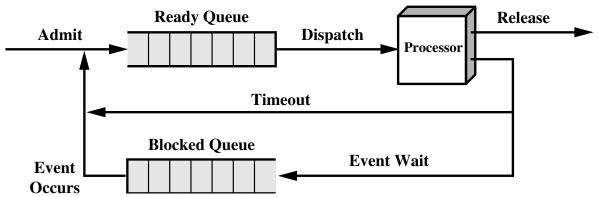
Slide 14

Non-running Threads

- Many separate reasons for a thread not running
  - another thread is running on the CPU
  - thread is blocked (waiting for an event)
  - thread is in initialisation phase (during creation)
  - thread is being cleaned up (during exit, kill)
- Dispatching ought to be fast
  - Shouldn't search through all threads to find runnable one
  - Achieved by distinguishing more thread states

Slide 15

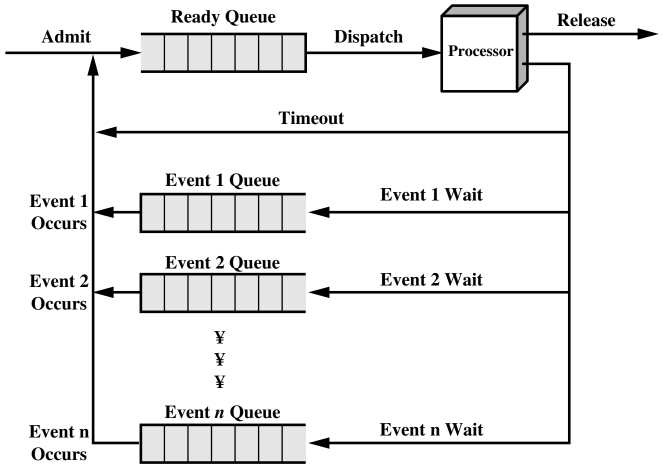
Separate queues



- Simplifies scheduler's job
- How about wakeup of blocked thread when event occurs?

Slide 16

Multiple wait queues:



Slide 17

Cooperative vs. Preemptive Multithreading

Cooperative multithreading:

- Threads determine exact order of execution
- Use yield() to switch between threads
- Problems if thread doesn't yield (e.g., buggy)

Preemptive multitasking:

- OS preempts thread's execution after some time
- Only guaranteed to work if H/W provides timer interrupt
- Implies unpredictable execution sequence!
  - thread switch can happen between any two instructions
  - threads may require concurrency control

Slide 18

User-level Operations on Threads in OS/161

- Start a new thread in OS/161

```
thread_fork(const char *    name,
           void       *    data1,
           unsigned long   data2,
           void      (*    func)(void *, unsigned long),
           struct thread **ret);
```

- Terminate thread
  - thread_exit()
- Yield CPU
  - thread_yield()
- Synchronisation:
  - thread_sleep(const void *addr)
  - thread_wakeup(const void *addr)

Slide 19

Processes and Threads

The OS stores information about Threads and Processes in Thread Control Block (TCB) and Process Controll Block (PCB)

- PCBs stored in process table
- TCBs stored in thread table

	Process	Thread
Address Space	✔	
Registers		✔
Program Counter		✔
Stack		✔
Open Files	✔	
State		✔
Signals and Handlers	✔	
Accounting Info	✔	✔
Global Variables		