Processes: A System View

CS 241

February 15, 2012

Copyright ©: University of Illinois CS 241 Staff
Today

Concurrency & Context Switching

Process Control Block
  • What's in it and why? How is it used? Who sees it?

5 State Process Model

MP2 Awards
Concurrenty

What is a sequential program?
  • A single sequence of control that executes one instruction at a time
  • Use system()

What is a concurrent program?
  • A collection of autonomous sequential programs, executing (logically) in parallel
  • Use fork()
What is fork good for?

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    pid_t pid;
    int i;

    if(pid = fork()) { /* parent */
        parentProcedures();
    }
    else { /* child */
        childProcedures();
    }

    return 0;
}
```
What is fork good for?

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
    pid_t pid;
    int i;

    while (1) {
        waitForClients();
        if (pid = fork()) { /* parent */
            resetServer();
        } else { /* child */
            handleNewClient();
        }
    }

    return 0;
}```
Why Concurrency?

Natural application structure
  • The world is not sequential!
  • Easier to program multiple independent and concurrent activities

Better resource utilization
  • Resources unused by one application can be used by the others

Better average response time
  • No need to wait for other applications to complete
Benefits of Concurrency

<table>
<thead>
<tr>
<th>No Concurrency</th>
<th>With Concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyboard</strong></td>
<td><strong>Wait for input</strong></td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Disk</strong></td>
<td></td>
</tr>
</tbody>
</table>

Wait for input

Input

Time
Benefits of Concurrency

<table>
<thead>
<tr>
<th>No Concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1</td>
</tr>
<tr>
<td>Client 2</td>
</tr>
<tr>
<td>Client 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>With Concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1</td>
</tr>
<tr>
<td>Client 2</td>
</tr>
<tr>
<td>Client 3</td>
</tr>
</tbody>
</table>

Wait for input

Input

Time
On a single CPU system...

Only one process can use the CPU at a time
  • Uniprogramming: only one process resident at a time
  • But we want the appearance of every process running at the same time

How can we manage CPU usage?
  • “Resource Management”
On a single CPU system...

Your process is currently using the CPU

```java
long count = 0;
while (count >= 0)
    count++;`n```

What are other processes doing?
On a single CPU system...

Answer
  • Nothing

What can the OS do to help?
  • Naively… Put the current process on ‘pause’

What are our options?
How can the OS help share CPU time?

Time slicing
• Use a hardware timer to generate a hardware interrupt

Multiprogramming
• Multiple processes resident at a time
• Wait until the process issues a system call
  ▪ e.g., I/O request

Cooperative multitasking
• Let the user process yield the CPU
Time slicing

A process loses the CPU when its time quantum has expired

Advantages?

Disadvantages?

```java
long count = 0;
while(count >=0)
  count ++;
```
Multiprogramming

Wait until system call

```c
long count = 0;
while(count >=0) {
    printf("Count = %d\n", cnt);
    count ++;
}
```

Advantages?

Disadvantages?
Cooperative Multitasking

Wait until the process gives up the CPU

```java
long count = 0;
while(count >=0) {
    count ++;
    if(count % 10000 == 0)
        yield();
}
```

Advantages?

Disadvantages?
Context Switch

Overhead to re-assign CPU to another user process

What activities are required?
Context Switch

Overhead to re-assign CPU to another user process

- Capture state of the user's processes so that we can restart it later (CPU Registers)
- Queue management (e.g. put process on “waiting” queue)
- Accounting
- Scheduler chooses next process
- Run next process
2 State Model

Processes

- enter
- dispatch
- pause
- exit

not running

running
2 State Model

Processes

System
2 State Model

Processes

System

What information do we need to keep in the queue?
Process Control Block (PCB)

In-memory OS structure
• User processes cannot access it

Contents:
• Identifiers
  ▪ pid & ppid (process ID & parent process ID)
• Processor state information
  ▪ User-visible registers, control and status, stack
• Scheduling information
  ▪ Process state, priority, what event the process is waiting for, ...
PCB (more)

Contents (cont’d):

• Inter-process communication
  ▪ Signals
• Privileges
  ▪ CPU instructions, memory
• Memory Management
  ▪ Segments, VM control 'page tables'
• Resource ownership and utilization
Five State Process Model

“All models are wrong. Some Models are Useful”

• George Box, statistician

2 state model

• Too simplistic
• What does “Not Running” mean?

7 state model

• Considers suspending process to disk
• See Stallings 3.2

Next: 5 state model
Malloc Contest Awards!