Computer System Scribe Notes Lecture 05 Section 01

Kunjan Bikram Kshetri

10 October 2010

Abstract

Scribe Notes for the Lecture 05 Part I. We recap the Deadlock prevention strategies and begin Memory management.

1 Deadlock Prevention

1.1 Banker's Algorithm

Banker's algorithm is a deadlock prevention algorithm developed by Dijkstra. It simulates the allocation of resources and then checks to test for possible deadlock conditions. This is run by the Kernel whenever process request is made.

Let us define our data structures.
Process : {P1, P2, P3.. Pn}
Resource : {R1, R2, .. Rn}

Available : {1... m }
Represents the number of resources available

Max{1..n}{1...m}
pi,rj : Represents the maximum number of a resource rj a process pi can ever request

Allocation{1..n}{1...m}
pi,rj :Represents the number of rj that pi is using right now at this point of time

Need{1..n}{1...m}
pi,rj : Represents the Max-Allocation, future requests or the outstanding requests

1.1.1 Safety Check

Resources are granted if the request is less than or equal to the Max and less than or equal to the available. Since the kernel doesn't know how much the process could eventually use, it looks at the Max reqest a process could make.

```
Safety-Check()
```

```
$Finish\{1..N\} \leftarrow False$
$\forall i. : Finish\{i\} \equiv false \& Need\{i\} \leq Work$
$Work \leftarrow Work + Allocation\{i\}$
$Finish\{i\} \leftarrow true$
**if** $\forall i.Finish[i] \equiv true$
  **then** "$Safe$"
```

When a process requests a resource, run the safety check, if it's safe to allocate, then only allocate. Otherwise put the process on the wait.

2 Deadlock Detection

If you can't prevent or avoid a deadlock, detect the deadlock and come out of it by either killing the process. Many OSes do not do this. However a simple way to detect deadlocks is to detect cycles in the dependency cycle of processes.

3 Memory Management

The memory gets accessed through CPU instructions such as LOAD and STORE and PC. What are we going to do as a memory manager?

3.1 Memory Isolation

Each processes touches his own memory. For example, we do not want Firefox to wreak havoc to Email Client.

Let us assign BASE address LIMIT to demarcate the section of memory a process has access to. These two numbers are stored in Base register Limit register , setting of which is a priviledge instruction.

The Kernel however needs access to the entire memory, which is why we set the base =0 and limit to everything else so that it can write stuffs to everything.

How do we make the program access its own memory?

1. One idea could be that at the compile time we asign absolute values to all the memory references. But this is very inflexible. If we change anything, we would need to recompile the entire program. Also no small instances of the program could run.
2. Another idea could be that we assign the values when the program is loaded from disk.
3. Yet another could be that we do the assignment in the execution time. That is for every memory request we do translation.

3.2 Logical Addresses

We introduce the notion of Logical addresses which is the memory address that a process uses while execution and it's mapped to a physical address. The process thinks it has the entire memory and refers to its own memory segment. We use a hardware called the MMU to translate that to the physical memory location. We use the hardware so that the Kernel doesn't need to be bothered about doing all the memory tranlsation.

4 MMU

The is discussed in the second half of the lecture.