SUSTech CS302 OS Lab7 Report

Title: **Deadlock**

Student: Name: Lu Ning, ID: 11610310

Time: 2019 4 13

Experimental Environment: linux C++

Objective: Understand the reason of deadlock, and the solution of deadlock. Understand several algorithm about dealing with deadlock, such as Banker's algorithm.

Deadline: **11:59 AM, 2019-04-17**

Summit by: Blackboard

Task：

Task 1. Implement the Banker's algorithm.

Task 2. Finish the report.

Experiments:

1. fundamental：

- What is deadlock？

  Situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource

- What are the requirements of deadlock？

  Circular wait.

  Mutual Exclusion.

  Hold and wait.

  No preemption.

- What's different between deadlock prevention and deadlock avoidance？

  Prevention:

  We can prevent deadlock by eliminating any of the above four condition.

  Avoidance:

  The system dynamically considers every request and decides whether it is safe to grant it at this point, considering the overall potential use of each resource for each process.

- How to prevent deadlock？Give at least two examples.

  Eliminate no preemption: We can allow high-priority process to preempt the resources.

Eliminate circular wait: Each resource will be assigned with a numerical number. A process can request the resources only in increasing order of numbering.

- Which way does recent UNIX OS choose to deal with deadlock problem, why？

  Ignore the problem and pretend that deadlocks never occur in the system.

2. Banker’s algorithm

- What data structures you use in your implementation？Where and why you use them？Are they optimal for your purpose？

Data structure:

int m : the number of resources’ kind

vector<int> : length m

maximum amount of instances provided by each kind of resources

current available amount of instances of each kind of resources

map<int, bool>: length is the number of processes in the system

Key: process id

Value: the finished state of key process.

map<int, vector<int>> :

Max: maximum demand of one process on each kind of resources

Key: process id

Value: a vector of length m, represents maximum demand of one process on each kind of resources

Need: current needed amount of each resources according to each

Key: process id

Value: a vector of length m, represents currently available demand of one process on each kind of resources

Reason

Because the process id is not sequential and not determined, so I can’t store the demand of each process in two dimension array. So I use the map to map the process id to a vector that stores the information.

And because the number of processes in the system is dynamic, and I use a map to store the finished state of each process running in the system.

For the m-length vector, it can be replaced by an array, I choose the vector data structure for my personal preference.

Optimal？

Because the process in the system is dynamic, so I think the implementation is optimal.

Conclusion:

From this lab, I obtain the knowledge of the dead lock and implement the bank algorithm by myself, which is beneficial for my understanding of the IPC.

Submission:

-OS_Lab7_studentID (directory)

---OS_Lab7_report_studentID.pdf (pdf version report)

---banker.cpp (code file)

Zip the directory with the same name and submit it