Operating Systems

Not Project 3 Memory Management

Another

Completely

Heuristic

Operating

System

Speaker : Wei-Ting Lu

The Goal of Project 3

To run the following two programs concurrently:

1. test/matmult
2. test/sort

What's wrong with the above programs?

Both of the programs need more memory space!

What we need to do in project 3?

1. To implement page swapping
2. Running matmult and sort at the same time under the scheduling algorithm which you had implemented in project 2
3. No way to pass this project by simply modify the memory size in "machine"

Hints

File system – swap space

Design and maintain three tables to handle page swapping:

1. PageTable
2. FrameTable
3. SwapTable

Modify *excption.cc* to Catch PageFaultException

Design your own virtual memory manager

File system – swap space

How to create a disk as swap space?

1. swap = new SynchDisk in your kernel

How access the disk as the virtual memory?

1. kernel->swap->WriteSector
2. kernel->swap->ReadSector

More details?

1. Tracing *synchdisk.cc*
2. See other headers in /filesys/

Three page tables

PageTable

1. One page table per process.
2. Decide your virtual page number.

FrameTable

1. Record every physical page’s information.
2. Each frame represent one physical page.

SwapTable

1. Record every sector’s information in swap.
2. The number of entries in SwapTable is the same as the swap sectors.
3. Each entry represent one frame in the disk.

PageTable

For each entry in PageTable

```
TranslationEntry {
  unsigned int virtualPage; //virtual memory page
  unsigned int physicalPage; //if in physical memory
  bool valid; //if in physical memory
  bool use; //been used(read or write)
  bool dirty; //been modified
  };
```

FrameTable

For each entry in FrameTable

```
FrameInfoEntry {
  Bool valid; //if being used
  Bool lock;
  AddrSpace *addrSpace; //which process is using this page
  Unsigned int vpn; //which virtual page of the process is stored in this page
  };
```

SwapTable

For each entry in SwapTable

```
FrameInfoEntry {
   Bool valid; //if being used
   Bool lock;
   AddrSpace *addrSpace; //which process is using this page
   Unsigned int vpn; //which virtual page of the process is stored in this page
   };
   // same as the FrameTable
```

Page Replacement

Tracing *Addrspace.cc*

1. Load one page at a time
2. Select a page and swap out to SwapTable
3. Mapping virtual address to physical address
4. Invoke “executable->ReadAt(&(kernel->machine->
mainMemory[physical address]), sizeToLoadNow, inFileAddr)”

Address mapping

Physical Address =
pageTable[(virtual address / PageSize)].physicalPage * PageSize +
(virtual address % PageSize)

Virtual Memory Manager

```
Class MemoryManager {
  Int TransAddr(AddrSpace *space, int virtAddr);
  //return phyAddr (translated from virtAddr)
  Bool AcquirePage(AddrSpace *space, int vpn);
  //ask a page (frame) for vpn
  Bool ReleasePage(AddrSpace *space, int vpn);
  //free a page
  Void PageFaultHandler();
  //will be called when manager want to swap a page from SwapTable to
  //FrameTable and the FrameTable is full.
  };
```

Files

For the disk usage details

1. /filesys/synchdisk.h
2. /filesys/synchdisk.cc
3. Other files in /filesys (Optional)

For the swap space initialization

1. /userprog/userkernel.h
2. /userprog/userkernel.cc

For the table maintaining

1. /machine/machine.h and /machine/machine.cc
2. /machine/translate.h and /machine/translate.cc

Files

For the table maintaining,

1. /machine/machine.h and /machine/machine.cc
2. /machine/translate.h and /machine/translate.cc

For the loading of pages

1. userprog/addrspace.h
2. userprog/addrspace.cc

Report

1. Problem analysis
2. How you implement to solve the problem in Nachos
3. What scheduling methods you based
4. Experiment result and discussion
5. Extra effort or observation

Please saved as [Student ID]_NachOS_report.pdf
E.g. r123456789_NachOS_report.pdf

Hand in report

Code

tar zcvf b99xxxxxx.tar.gz ./nachos-4.0

Mail your code and report to TA before the deadline:

argonmisir@gmail.com