Operating Systems, Summer 2006

CS 5/43201
Department of Computer Science
Kent State University

Project#2: Due Date_____

Preparation: For this assignment you will need to copy the contents of the *subdirectories /userprog, /filesys, machine/* from *~javed/pub/nachos-3.4-hp/code* directory. Before you compile, open the **Makefile** and uncomment the two lines which looks like following and uncomment the rest.

```
cd filesys; $(MAKE) depend
cd filesys; $(MAKE) nachos
```

Now compile Nachos, go to the *filesys/* directory and test the following commands:

```
nachos –f
nachos  -l
nachos –cp switch.o switch.o
nachos -l
nachos –p /
nachos –cp  synchdisk.h synchdisk.h
nachos –p synchdisk.h
nachos -D
nachos –r switch.o
nachos –l
```

Like last assignment, you now need to study the source code. Also read the article "A Roadmap Through Nachos" by Tom Narten (it is in the class web site) very carefully. Also read the following source code files in the thread/ directory.

Nachos file system has a bitmap, a directory (root only), one file-header for each file (and the file data). The file management system of nacho has separate modules for each of these resources. It also has two manager modules one to perform the main management tasks and the other to support programs. It has the following components.

Disk.cc	routines which simulate the virtual disk machine. It is not a part of Nachos but of the virtual machine.
Bitmap.cc:	routines to manage a bitmap (used to keep track of free sectors in the DISK).
Directory.cc	routines to manage a directory table (Nachos have only one now).
Filehdr.cc	routines to manage the file header.
Syncdisc.cc	routines just above machine level which can be called to read and write from DISK without worrying for mutual exclusion.
Filesys.cc	routines which use the above and perform the management operations of Nachos File System.
Openfile.cc	routines which users programs can call to manage (open, seek, read, write, find out length etc.) open Nachos files.
Fstest.cc	A set of unix like application programs which tests the filesystem.

Road Map Questions: Now test your understanding by solving the following road map question. As before there is no grade reward for answering them

1. What is the size of one sector in a Nacho disk? How many such sectors are there? What is the size of Nachos DISK? Where are these defined?
2. Which sector stores the root directory?
3. Which sector stores the bit map to track free space?
4. What are the names of the macro variables which define the sector numbers in which the root directory and the bitmap that tracks free disk sector is stored?
5. Draw the structure of the bitmap.
6. In a typical disk some sectors may be damaged. Can you upgrade the bitmap which also can note for damaged bitmap?
7. Where and how the size of the root directory is set?
8. What is the currently defined limit on the files per directory?
9. Draw the structure of the root directory.
10. What is the command line option for formatting a directory?
11. Where are the codes for formatting a directory?
12. Why two sets of (total 4) WriteBack() operations are performed while formatting or initializing a filesystem?
13. What are the steps involved in creating a new file? Trace the routines called in creating a file.
14. Which routine allocates the sectors when a new file is created?
15. Try running “nachos –cp Makefile /Makefile”. Why this statement results in core dump?
16. What is the exact maximum file size that Nacho can store? How this limit is decided?
17. What is stored inside dataSectors?
18. Can you draw a fileheader structure?
19. What ByteToSector(int offset) in file filehdr.cc is doing?
20. Trace all the write routines called when the copy utility in fstest.cc writes the unix file into the Nachos DISK.
21. Why there are so many write routines? What is the specific role of each of these routines?
22. What is the role of variable “semaphore” in synchdisk.cc?
23. What is the role of variable “lock” in synchdisk.cc?
24. What is the purpose of synchdisk.cc routine?

Assignment:

In the last assignment we have studied the process synchronization mechanism of Nachos. In this project we will study and improve the file system of Nachos. We will start with a rudimentary Nachos which has the simplest conceivable implementation of a file system. In current version we will improve several features of this file system.

In this assignment you need to submit the modified source code files to TA. Therefore, before you modify a program, make a backup copy. Also in every place, where you add/modify the code, you must include a comment stab:

```
/*MODIFICATION BY your name DATE: date FOR: question number*/
```

This will help you as well as the TA to locate your modifications more easily. Also, on top of each file include your name, data and project number.

1) **Design Recovery:** The first assignment is to verify our understanding of the structure of nachos File System Manager. To do that we will draw a diagram of nachos File system. The eight routines you studied above together represent various components of Nachos file system, plus the virtual hardware, plus an application layer module which use the file system. Your assignment is to draw a block diagram. Use the following guideline. You will draw only one diagram. However, I will divide the grading based on your success in following each of these guidelines.

    a. Each block should represent one of these eight units. Name each blocks with no more than three words. Select the words very carefully. The name should capture the essence of their functionality. (40 points, knowing the components).

    b. Show the relationship between the blocks with arrows. If a routine in block A calls for any service from block B then there should be an arrow from block A pointing to B. NO need to draw multiple arrows between the same pair of blocks. Just showing ONE will suffice. If there are dependencies in both the directions, use ONE bi-directional ARROW. (60 points, knowing the dependencies).

    c. While drawing, place the hardware layer at the bottom and the application layer at the top to page. Try to organize the blocks in such a way that the arrows can be drawn either vertically or horizontally with minimum crossover. Obtain the neatest diagram that you can, from the findings of a. b. (50 points, discovering the structure).

    d. Write a short critique (no more than half a page) on the organization of Nachos file system. (Hint: Is it hierarchical? layered? Modular? Is there is cyclic dependency? Is it modular? Is it over complicated?) (50 points, appreciation of the design).

    Submission: Draw the diagram in a MS Word file Project2.doc.

2) **Implementation of Hieararchical Directory System:** Nachos file system is ‘flat”. It has only one directory “root”. All files are in this root directory. This assignment requires you to implement a hierarchical directory system.

    a. Implement command line options “–mkdir”, “-rmdir”. The commands “-r”, “-p”, and “-l” should now be able to accept full path names. The command “-D” still should print out the entire content of the filesystem in a readable manner. You can assume that the file and directory name arguments for all these commands will contain full or absolute path name. At this point you do not need to implement the concept of working directory. Every

filename will have absolute pathname with it (all starts with "\". Directories and seperated with "\"). (100 points).

b. Make your filesystem robust. So that if anybody inputs a wrong file name (such as it does not start with "\", or a path in invalid) an appropriate error message will be generated. (40 points)

c. Implement working directories. All file system commands should now be able to accept file names relative to current directory. The filesystem should be able to remember the current directory of a process. You also need to implement commands "-cd" to change the current directory, and "-pwd" to print out the current directory. Your code should begin searching at the root directory if file name begins with "\" otherwise it should start at the current directory. (60 points)

Submission: For each part of this assignment, Neatly describe your plan in the MS Word file Project2.doc and draw diagram if needed (20%). List one by one the key modifications made in your source code (10%). Also list the source code files you have modified (5%).

You also need to submit the modified source code so that TA can run them and test (70%). However, you do not need to submit three different sets of files for the above three parts of this assignment. Send only the latest one. To prepare for submission, copy each of the modified source code files by appending ".pb2" at the end of their name. (For example file "*filehdr.cc*" should be renamed as "*filehdr.cc.pb2*") and mail them to TA using procedure described later.

3) **Access Control:** Nachos has no security or access control for files. In this assignment we will implement a simplified unix like access control/permission technique for Nachos file. We will allow files to be either writable (read, write, and deletable) or read-only.

a. Implement the two command line options "-pwirte" and "-pread", which should be able to set the read/write permission of the specified file. (Formats are as following: "nachos -pwrite filename", and "nachos -pread filename"). (100 points)

b. Implement a command line option "-la" (similar to "-'l" option, which should list all the files in a specified directory along with the permission type of each file. (50 points)

Submission: For each part of this assignment, Neatly describe your plan in the MS Word file Project2.doc and draw diagram if needed (20%). List one by one the key modifications made in the source code (10%). Also list the source code files you have modified (5%).

You also need to submit the modified source code so that TA can run them and test (70%). However, you do not need to submit two different sets of files for the above two parts of this assignment. Send only the latest one. To prepare for submission, copy each of the modified source code files by appending ".pb3" at the end of their name. (For example file "*filehdr.cc*" should be renamed as "*filehdr.cc.pb3*").

4) **Larger File System:** Nachos files have a limit on the maximum file size which is close to 4K. The assignment is to upgrade the filesystem so that larger files can be stored.

a. What is the current maximum limit on file size? Explain how the current limit is imposed. Design an upgrade strategy so that the list that keeps track of the sectors in a file can be arbitrarily increased at the time of file creation. (50 points).

b. Modify the file system (specially "filehdr.cc" and filehdr.h") to implement your upgrade strategy. Create a test unix file test.txt which has size about 5K. Test your system by copying this file into the modified file system. (150 points)

Submission: For each part of this assignment, Neatly describe your plan in the MS Word file Project2.doc and draw diagram if needed (20%). List one by one the key modifications you have made in your source code (10%). Also list the source code files you have modified (5%).

You also need to submit the modified source code so that TA can run them and test (70%). To prepare for that copy each of the modified source code files by appending ".pb4" at the end of their name. (For example file "filehdr.cc" should be renamed as "filehdr.cc.pb4") and mail them to TA as described later.

How to Submit All:

In this assignment you have created a set of program files ***.pb*** and one answersheet Project2.doc which contains all your explanations and your answer to Q#1. Submit a hardcopy of Project2.doc to me in the class on project due date. You also need to submit the modified source code files to TA. Before submitting, in every place when you have modified the code include a comment stab:

```
/*MODIFICATION BY your name DATE: date FOR: question number*/
```

This will help TA to locate your modifications more easily. Also, on top of each file include your name, date and project number. Add:

```
/************************************************
Name:
Date:
Project/Question Number:
OS CS 5/43201, Summer 98
Instructor: KHAN, KSU
************************************************/
```

For source files (*.cc) comment them.

You now need to mail all of the modified source code files into one package using the following procedure:

%shar *.pb2 *.pb3 *.pb4 Project2.doc > shar.project2
%elm –s "Project 2 for type your name here" xzou1@kent.edu < shar.project2

Check thoroughly before you submit. If you need to re-send, for any reason inform TA (xzou1@kent.edu) beforehand. Keep a copy of all the files including shar.project1 in your directory. Do not modify them afterward. If need arises, TA may want to check these files. Any modification afterward (reflected in the file date) will result in late submission penalty.

Grading:
See notes to grader in the website.

Cheating and Copy:
Projects have to be done individually. If a copy is caught, all involved submissions (original as well as the copies) will be penalized. So it is your responsibility to guard your work. Secure the read/write access of your directories. Any copy will result in ZERO grade for the assignment for both party. Only exception is when you report the theft of your work in advance.