NACHOS PROJECT ASSIGNMENT 3

CS 370 – OPERATING SYSTEMS, LUMS

"THIS ASSIGNMENT HAS BEEN ADAPTED FROM CS162 COURSE @WASHINGTON-UNIVERISTY"

USERLEVEL PROGRAMS & MULTIPROGRAMMING

DUE DATE: REFER TO INDIVIDUAL TASKS*

* Slips not applicable on non-programming tasks.

APPENDIX 1

The new code is spread over several directories. There are some new kernel files in "userprog", there are a few additional machine simulation files in "machine", and a stub file system in "filesys". The user programs are in "test", and utilities to generate a Nachos loadable executable are in "bin". Since Nachos executes MIPS instructions (and there aren't very many MIPS machines left!), we also provide you a cross-compiler. The cross-compiler runs on Linux and compiles user programs into MIPS format.

You may find Narten's "road map" to Nachos helpful; Google it. Also, it is OK to change the constants in "machine.h", for example, to change the amount of physical memory, if that helps you design better test cases (you may choose not to). The files for this assignment include:

USERKERNEL.H, USERKERNEL.CC -- routines for booting and testing a multiprogramming kernel.

ADDRSPACE.H, ADDRSPACE.CC -- create an address space in which to run a user program, and load the program from disk.

SYSCALL.H -- the system call interface: kernel procedures that user programs can invoke.

EXCEPTION.CC -- the handler for system calls and other user-level exceptions, such as page faults. In the code we supply, only the "halt" system call is supported.

BITMAP.H, BITMAP.CC -- routines for manipulating bitmaps (this might be useful for keeping track of physical page frames)

FILESYS.H, OPENFILE.H (found in the filesys directory) -- a stub defining the Nachos file system routines. For this assignment, we have implemented the Nachos file system by directly making the corresponding calls to the UNIX file system; this is so that you need to debug only one thing at a time.

TRANSLATE.H, TRANSLATE.CC -- translation table routines. In the code we supply to run "halt", we assume that every virtual address is the same as its physical address -- this restricts us to running one user program at a time. You will generalize this to allow multiple user programs to be run concurrently. We will not ask you to implement virtual memory support until in assignment 3; for now, every page must be in physical memory.

MACHINE.H, MACHINE.CC -- emulates the part of the machine that executes user programs: main memory, processor registers, etc.

MIPSSIM.H, MIPSSIM.CC -- emulates the integer instruction set of a MIPS R2/3000 processor.

CONSOLE.H, CONSOLE.CC -- emulates a terminal device using UNIX files. A terminal is (i) byte oriented, (ii) incoming bytes can be read and written at the same time, and (iii) bytes arrive asynchronously (as a result of user keystrokes), without being explicitly requested.

SYNCHCONSOLE.H, SYNCHCONSOLE.CC -- provides synchronized access to the console device.

Some text in this handout has been borrowed from Md Tanvir Al Amin's document on MULTIPROGRAMMING, PROCESS MANAGEMENT AND CONSOLE that may be found on this link:

http://tanviramin.com/documents/nachos2.pdf

Students may read this document for further understanding of this assignment in particular and NACHOS in general.