Homework 1
Computer Systems Organization II, V22.0202.003 Spring 2005, Professor Yap

Due: Tue Feb 1, in class

INSTRUCTIONS:

- Please read questions carefully. When in doubt, please ask.

---

1. (16 Points)
   Answer Questions 1 to 4 of [Tanenbaum, p.67]

2. (14 Points)
   Question 11 of [Tanenbaum, p.68]. Speed of various memory levels.

3. (40 Points)
   This question assumes that you have downloaded Cygwin, gcc, make and tar. Please look at our class links for this information.

   Enclosed is a simple program called `fork.c` and `test.c` that calls fork() and exec*(). They are put into a tar file called "homework1.tar". You are to write a variation of these programs.

   - 1. Untar the file homework1.tar (move the file to a directory and type "tar -xvf homework1.tar"). Then Please compile and run them using gcc.
   - 2. Please create a variation of these programs that creates an arbitrary number of processes until your system crashes. For safety, we suggest stopping all other programs before you run the test. A similar question is Question 28 of [Tanenbaum, p.69].
   - 3. There is a simple make program also provided here. You must use this to help you programming and testing.
   - 4. Read the following submission rules carefully, as up to 10 percent of this question will deducted for non-compliance.
     – Make sure that your program is well-documented.
     – Make sure your name and other information is clearly written in the program.
     – ANY ACKNOWLEDGEMENT OF SOURCES SHOULD BE EXPLICITLY MADE HERE. Otherwise, it is considered plagiarism.
     – Your program file name MUST be called h1.c.
     – You must put h1.c, Makefile, and any other necessary information into a tar file called h1.tar. (Type "tar cvf h1.tar h1.c Makefile ..." to create this tar file).
     – Send the tar file to me. After I untar your file, I expect to type "make" to compile it, and ti type "make test" to run the compiled program.

```
// file: test.c
// Comp.Sys.Org.II, Spring 2005, Yap.
// The following code is a simple modification of the "fork.c" code from
//
// Dr Ian G Graham, Department of Information Technology, GUGC
// Copyright  2000-2003
//
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <sys/wait.h>

extern int errno;     /* system error number */

void syserr(char* );  /* error report, abort routine */

int main(int argc, char *argv[])
{
   pid_t pid;
   int rc;

   pid = getpid();
   printf("Process ID before fork: %d\n", pid);

   switch (fork()) {
      case -1:
         syserr("fork");
      case 0:

/* execution in child process */
         pid = getpid();
         printf("Process ID in child after fork: %d\n", pid);
         execlp("echo", "0", "Child to world: Hello!",NULL);
         syserr("execl");
   }

/* continued execution in parent process */

   pid = getpid();
   printf("Process ID in parent after fork: %d\n", pid);

   //getchar(); // supposed to hang until I type something
   exit(0);
}

void syserr(char * msg)
{
   fprintf(stderr,"%s: %s", strerror(errno), msg);
   //abort(errno); // variant
   abort();
}
```