# Sistemas Operativos - Práctica 4

Multiprogramación en Nachos

La segunda fase de Nachos es soporte para multiprogramación. Como en la primera parte, ya se provee un poco del código necesario. Se debe completarlo y mejorarlo.

Puede ser útil cambiar algún parámetro de la máquina para desarrollar mejores casos de prueba (por ejemplo la cantidad de memoria física). Sin embargo conviene tener siempre presente que el código dentro del directorio ``machine" representa la máquina física sobre la que se ejecuta el SO, por eso en general no se debe modificar el código dentro de ese directorio.

1. Implementar las llamadas al sistemas y la administración de interrupciones. Se deben soportar todas las llamadas al sistema definidas en syscall.h, exceptuando fork y yield. Para poder probar `exec' y `wait' se debe realizar primero el punto 2.

   Notar que la implementación debe ser ``a prueba de balas", o sea que un programa de usuario no debe poder hacer nada que haga caer el sistema operativo (con la excepción de llamar explícitamente a ``halt").

   Para la implementación del las llamadas que acceden a la consola probablemente sea útil implementar una clase ``SyunchConsole", que provea la abstracción de acceso sincronizado a la consola. En ``progtest.cc" está el comienzo de la implementación de SynchConsole. La clase de acceso sincronizado a disco (SynchDisk) puede servir de modelo.

2. Implementar multiprogramación con rebanadas de tiempo (time-slicing). Será necesario:
   1. Proponer una manera de ubicar los marcos de la memoria física para que se puedan cargar múltiples programas en la memoria (ver bitmap.h).
   2. Proveer una forma de copiar datos de/desde el núcleo al espacio de memoria virtual del usuario (ahora las direcciones de memoria que ve el programa de usuario no son las mismas que ve el núcleo)
   3. Forzar cambios de contexto después de cierto número de tics del reloj. Notar que, ahora que está definido USERPROG, scheduler.cc almacena y recupera el estado de la máquina en los cambios de contexto.

3. La llamada ``exec" no provee forma de pasar parámetros o argumentos al nuevo espacio de direcciones. UNIX permite esto, por ejemplo, para pasar argumentos de línea de comando al nuevo espacio de direcciones. Implentar esta característica.

   Hay dos forma de hacerlo, una es al estilo de UNIX: copiar los argumentos en el fondo del espacio de direcciones virtuales de usuario (la pila) y pasar un puntero a los argumentos como parámetro a main, usando r4 para pasar el puntero. La otra forma es agregar una nueva llamada al sistema, que cada espacio de direcciones llama como primera cosa en main y obtiene los argumentos para el nuevo programa.

4. Escribir un intérprete de comandos sencillo y al menos dos programas utilitarios (como ``cat" y ``cp"). El intérprete lee un comando de la consola y lo ejecuta invocando a la llamada ``exec".

---