# Sistemas Operativos II - LCC
# Práctica 3 - 2013.
# Programas de usuario y Multiprogramación
# *Entrega:* 22 de Mayo.

**NOTA:** Debe utilizar el SVN de la materia creando un subdirectorio por alumno/grupo en:
`https://dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Alumnos/2013/`

## 1. Programas de usuario y 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. Al ejecutar programas de usuario, tendremos **dos espacios de direcciones**. El primero es que utiliza el kernel de nachos el cual corre en la máquina x86. El segundo es el del proceso de usuario que corre **sobre la máquina MIPS simulada**, por lo tanto serán direcciones en la máquina simulada. Luego los punteros (arreglos y strings) no pueden ser intercambiados entre estos dos espacios de direcciones.

   Proveer una forma de copiar datos desde el núcleo al espacio de memoria virtual del usuario y viceversa. Debe tener también dos tipos de funciones, una que lea/escriba strings terminadas por cero y otra que lea N bytes. Estas funciones son:

   ```
   void readStrFromUsr(int usrAddr, char *outStr);
   void readBuffFromUsr(int usrAddr, char *outBuff, int byteCount);
   void writeStrToUsr(char *str, int usrAddr);
   ```

```
void writeBuffToUsr(char *str, int usrAddr, int byteCount);
```

Para acceder a la memoria del usuario puede usar las funciones `Machine::ReadMem` y `Machine::WriteMem`.

2. 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 3. Se sugiere implementarlas en el siguiente orden: Create, Read/Write (a la consola inicialmente), Open, Close, Exit, Join, Exec.

   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 "SynchConsole", 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. Tenga en cuenta que a diferencia del disco, en este caso un hilo queriendo escribir no debería bloquear a un hilo queriendo leer.

3. Implementar multiprogramación con rebanadas de tiempo (time-slicing). Será necesario:

   - Proponer una manera de ubicar los marcos de la memoria física para que se puedan cargar múltiples programas en la memoria (ver bitmap.h).
   - 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.

4. 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 la cantidad y r5 para pasar el puntero. Esta forma puede ser vista en la Figura 1.

   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.

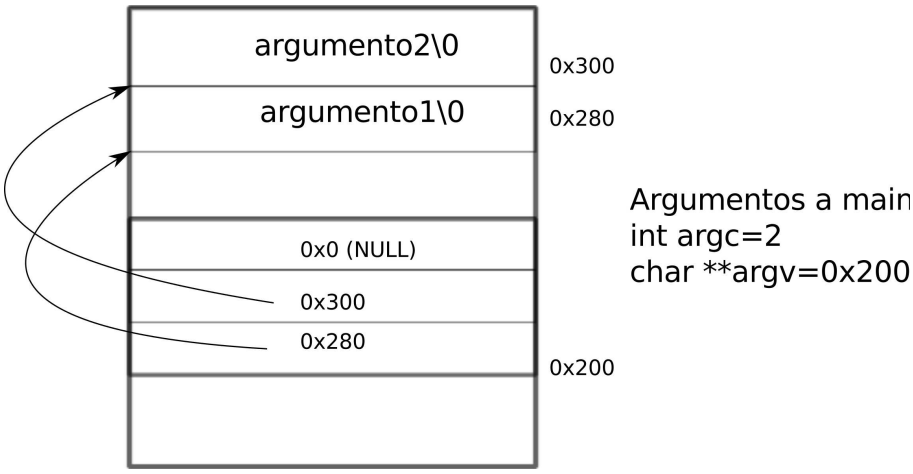


Figura 1: Layout de la pila al pasar argumentos a main

5. 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”. Si el comando comienza con el caracter “&” el comando debe ser ejecutado en background.