## Curso de Posgrado No Estructurado

# "Implementación de Sistemas Operativos"


### Docentes: Lic. Edgardo Hames, Lic Daniel Moisset, Lic. Nicolás Wolovick

### Fa.M.A.F. - U.N.C.


### Primer Semestre 2011


**Contacto: nicolasw{arroba}famaf.unc.edu.ar**


## Introducción

Durante los 8 años de dictado del taller de la materia "Sistemas Operativos" de la Lic. en Ciencias de la Computación, los Lic. Hames, Lic. Moisset y Lic. Wolovick han visto que la mejor forma de encarar un Laboratorio para dicha materia sería dar alguno de los proyectos educativos para la construcción de un pequeño Sistema Operativo de corte educativo. Existen muchos de estos proyectos, todos ellos de gran calidad, tanto en su parte técnica como en su parte pedagógica.
Para nombrar solo algunos:

- Nachos, de la Universidad de Washington,
- GeekOS, de la Universidad de Maryland,
- JOS, del MIT.

Creemos que un curso de posgrado en esta línea será muy beneficiosa para quienes tienen intereses muy marcados en el estudio de la capa más baja de toda la pila de software que compone una computadora: el Sistema Operativo.

Nos motivan tres ideas:

1. El Sistema Operativo es exactamente donde el software y el hardware se encuentran.
2. Existen problemas abiertos importantes a nivel académico en el área de bajo nivel, y es importante estimular a los alumnos con aptitudes e intereses en ese área.
3. Se aprende haciendo.

El primer punto se ve claramente en currículas donde los temas de Sistemas Operativos se dan en una materia que se denomina *Arquitectura de Computadoras 2*.
Para apoyar el segundo solo hace falta mencionar la importancia tanto para la Academia como para la Industria de Conferencias como USENIX y SIGOPS.

El proyecto a seguir será **GeekOS**, cuyo paper seminal es "Running on the Bare Metal with GeekOS", presentado en 2004 en la conferencia de la ACM sobre Educación en Ciencias de la Computación (SIGCSE).

La propuesta de GeekOS está armada alrededor de seis proyectos que **incrementalmente** construyen el Sistema Operativo.
Los proyectos son los siguientes:

- Proyecto 0: "`hello world`" en un hilo de kernel.
- Proyecto 1: carga de un ejecutable en formato ELF, generado por compiladores estándar.
- Proyecto 2: implementación de procesos de usuario utilizando segmentación de memoria; implementación de llamadas a sistema.
- Proyecto 3: implementación de un planificador alternativo y medición de su desempeño; implementación de primitivas de sincronización por semáforos.
- Proyecto 4: implementación de memoria virtual (paginación).
- Proyecto 5: implementación de un sistema de archivos tanto en su disposición en disco como el Virtual Filesystem Switch.

# Contenidos

La propuesta del curso consiste en la realización individual y guiada de los seis proyectos. Para sostener estos desarrollos, cada proyecto tendrá el apoyo de 2 o 3 clases teóricas de 2hs de duración.

- Proyecto 0:
    1. Descripción general de la materia. Compilación utilizando herramientas modernas y ejecución bajo el emulador Bochs.
    2. Code walkthrough del proyecto.
- Proyecto 1:
    1. Formato de ejecución y enlazado (ELF).
    2. Del código fuente a la ejecución: descripción de convenciones y formatos.
    3. Bootloading.
    4. Vulnerabilidades comunes que aparecen en el parseo del formato ELF.
- Proyecto 2:
    1. Segmentación en la arquitectura ia32 de Intel. Modelo de memoria de un proceso.
    2. Implementación de llamadas a sistema.
    3. Pruebas de procesos de usuario maliciosos: vulnerabilidades en la segmentación y en las llamadas a sistema.
- Proyecto 3:
    1. Planificadores multilevel feedback, métricas para el análisis de desempeño.
    2. Implementación de primitivas de sincronización: semáforos.
    3. Workloads para el análisis de desempeño de las políticas de planificación.
- Proyecto 4:
    1. Memoria virtual usando paginación y segmentación.
    2. Alocación dinámica de páginas, políticas de reemplazo de páginas.
    3. Implementación de la userheap a través de la llamada a sistema brk().
    4. Programas userspace que ejercitan los mecanismos y políticas de la memoria virtual.
- Proyecto 5:
    1. El Virtual Filesystem Switch (VFS). Puntos y tabla de montaje.
    2. Formato de sistemas de archivos en disco: PFAT y GOSFS (GeekOS FileSystem).
    3. Buffercache. Concurrencia y disciplinas de locking para el FS.
    4. Pruebas industriales para Sistemas de Archivos.

El resto de las 120hs que componen el curso serán de **trabajo individual supervisado**, ya sea de manera personalizada con la realización de code-walkthroughs y code-sharings, asi como de manera indirecta inspeccionando los repositorios de código (SVN) que serán puestos a disposición de los estudiantes.

Se deberá realizar también un **Trabajo Final**.
Dicho trabajo deberá ser consensuado con los docentes y para ser aprobado, deberá mostar madurez en la problemática de Implementación de Sistemas Operativos.
A manera de ejemplo, posibles Trabajos Finales podrían ser portar todo lo realizado durante el curso a otra arquitectura, como por ejemplo ARM o MIPS. Este trabajo, que se resume en pocas palabras, involucra una comprensión profunda de la nueva arquitectura asi como de todos los proyectos realizados.
Otra posibilidad consiste en continuar con la reescritura que se está realizando desde cero por su creador David Hovemeyer, a fin de darle más modularidad y claridad al código.

# Mecánica de Trabajo

El dictado de clases estará guíado por el inicio y fin de cada proyecto.
Por cada proyecto se dará una clase inicial donde se indicarán los objetivos y forma de trabajo.
En las clases sucesivas, se darán los temas correspondientes a cada proyecto, además de las consultas propias que surgen en el desarrollo.
Una vez finalizado el período de trabajo, se realizará una **puesta en común** y discusión de lo trabajado por cada uno de los estudiantes, y se obtendrá una **implementación patrón** que podrá surgir de uno o más implementaciones, y ésta servirá para obtener un punto de sincronización en el inicio del nuevo proyecto.

Dado un tiempo de cursado de 14 semanas, con dos clases por semana se propone el siguiente cronograma tentativo:

Proyecto 0: 1 sem.
Proyecto 1: 2 sem.
Proyecto 2: 3 sem.
Proyecto 3: 2 sem.
Proyecto 4: 3 sem.
Proyecto 5: 3 sem.
Total: 14 semanas.

Dentro de estos tiempos se contabilizan:

- Inicio del proyecto mostrando los alcances y dificultades
- Clases con material relevante para la resolución de los proyectos.
- Clases de consulta para fortalecer los avances.
- Clase para la puesta en común y la elección de la implementación patrón para continuar con el proyecto siguiente.

## Requisitos e Inscripción

El alumno deberá tener formación comprobable, no necesariamente formal en Sistemas Operativos, Arquitectura de Computadoras, y un nivel avanzado en programación de Lenguaje "C".
Las inscripciones se realizan en la Secretaría de Posgrado de la Fa.M.A.F., a partir del 14 de Febrero de 2011.
Los alumnos que no pertenecen a la Fa.M.A.F. deberán presentar copia del título y certificado analítico.

## Modalidad de Dictado y Evaluación

Habrá un encuentro por semana.
Las clases teóricas serán dictadas por los Lic. Hames, Moisset y Wolovick.
Además se **invitará a dar charlas** a expertos locales en temas de compilación, seguridad y formato de ejecutables.

La materia tiene un fuerte contenido práctico, es por ello que no habrá evaluaciones parciales, la materia será **únicamente promocional**.
La metodología de evaluación consiste en la realización de los **seis proyectos**, y del **trabajo final**.
El esquema de calificación le asignará **60% a la nota de proyectos** y **40% al trabajo final**.

## Carga Horaria

Se contabilizan 30hs de dictado de clases frente a alumnos, 30hs de consulta y 60hs de trabajo supervisado, totalizando **120hs** de clase.

## Infraestructura

- Repositorio SVN individual para cada alumno dado por la Facultad.
- Software estandar: bochs-2.4.2, gcc-4.4, editores de texto para programación (genie, kedit, vim, emacs, etc.).
- Espacio de disco duro para albergar imágenes de disco de 32MB.
- Cañon retroproyector para la inspección colectiva de códigos y dictado de clases.

## Bibliografía


- David Hovemeyer, Bobby Bhattacharjee, Jeffrey K. Hollingsworthl, *Running on the bare metal with GeekOS*, Proceedings of the 35th SIGCSE technical symposium on Computer science education (SIGCSE-04), SIGCSE Bulletin, Vol. 36, 1, pp. 315-319, ACM Press, March 3-7 2004.
- Wayne A. Christopher, Steven J. Procter, Thomas E. Anderson, *The Nachos Instructional Operating System*, USENIX Winter, pp. 481-488, 1993.

- Geek&Poke, *About Coders*, Lulu Editions, 2010.
- Tom Shanley, *Protected Mode Software Architecture*, MindShare Inc., 1996.
- TIS Committee, *Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification*, Version 1.2, May 1995.
- Intel Corp., *IA-32 Intel® Architecture Software Developer's Manual, Volume 3: System Programming Guide*, June 2005.
- Intel Corp., *Intel® 64 and IA-32 Architectures, Software Developer's Manual Volume 2A: Instruction Set Reference, A-M*, June 2010.
- Intel Corp., *Intel® 64 and IA-32 Architectures, Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z*, June 2010.
- Dominic Giampaolo, *Practical File System Design*, Morgan Kaufmann, 1998.
- Steve D. Pate, *UNIX Filesystems: Evolution, Design, and Implementation*, Wiley, 2003.