UNIVERSIDAD NACIONAL DE LA MATANZA
Departamento de Ingeniería
e Investigaciones Tecnológicas

Cátedras de:

Sistemas de Computación II (Plan 1997)

Sistemas Operativos (Plan 2009)

Jefe de Cátedra: Fabio E. Rivalta
Asesor académico: Carlos Neetzel
Equipo de Docentes: Boettner F., Catalano L.,
de Lizarralde R, Álvarez R., Volker M., Villamayor A
Auxiliares docentes: Loiacono F., Gariup G, Alessandelo, G., Hirschfeldt D.,
Rodriguez A., Piubel F., Barillaro, S., Barbieri D., Bobr, G., Moscardi, E.

PLANIFICACIÓN Y GUÍA DE TRABAJOS PRÁCTICOS (Segundo cuatrimestre 2014)

Contenidos:

- Reglamento, Programa y Planificación.
- Comentarios Previos
- Introducción al Diseño de Software.
- Guía de Trabajos Prácticos Grupales.

CÓDIGO ASIGNATURA
618 (Plan 1997)
1115 (Plan 2009)

DEPARTAMENTO: ***Ingeniería e Investigaciones Tecnológicas***

ASIGNATURAS: **Sistemas de Computación II**
Sistemas Operativos

1. OBJETIVOS.

1.1. OBJETIVOS DEL CURSO.

- Brindar los conceptos fundamentales y su respectiva actualización tecnológica sobre los Sistemas Operativos.
- Facilitar una actualización sobre las terminologías, y desarrollos tecnológicos sobre Sistemas Operativos Modernos.

1.2. OBJETIVOS DE APRENDIZAJE.

- Que el alumno adquiera el dominio de conceptos básicos y actualizados sobre los Sistemas Operativos e introducir los lineamientos generales de nuevos desarrollos tecnológicos en estos temas.
- Generar una concepción global y un enfoque selectivo para las soluciones algorítmicas de los diferentes problemas que ocurren dentro de un computador y la correcta utilización del mismo.

1.3. META OPERATIVA:

- Se tratara que el alumno al finalizar la materia logre:
- Adquirir el vocabulario y usarlo con precisión.
- Conocer en forma amplia y general la misión y funcionamiento de los componentes de los Sistemas Operativos de un computador.
- Analizar y evaluar por si mismo un Sistema Operativo de cualquier equipo existente en plaza.
- Desarrollar en el Alumno, el interés por la investigación, usando libros y publicaciones propuestas por el Docente.
- Crear en el Alumno, una capacidad de resolución de problemas mediante una adecuada ejercitación práctica.
- Motivar en los alumnos a proponer algunos temas de interés para desarrollar o investigar o encontrar diferentes soluciones a los mismos.

2. ALCANCES.

Los temas a tratar contemplarán básicamente los módulos temáticos que propondrá la cátedra. Su profundidad abarca la extensión de todos los temas específicamente mencionados en el Programa de SISTEMAS DE COMPUTACIÓN II (Plan 1997) y SISTEMAS OPERATIVOS (Plan 2009) vigentes.

PROGRAMA ANALÍTICO.

CONTENIDOS TEÓRICOS Y PRÁCTICOS DE LAS ASIGNATURAS: SISTEMAS DE COMPUTACIÓN II (Plan 1997) y SISTEMAS OPERATIVOS (Plan 2009) – VERSIÓN 2014

Módulo 1: Generalidades de los sistemas operativos

Conceptos de Arquitecturas CISC y RISC. Conceptos de Microprogramación. Conceptos del Lenguaje Assembler.
Interacción con el Sistema Operativo. Limitaciones del Hardware de las Computadoras.
Introducción a los SO. Clasificación. Conceptos fundamentales y conceptos básicos de SO.
Terminología, definiciones y funciones de SO.
Características comunes a todos los SO. Organización y estructura interna de los sistemas operativos. Componentes mínimos de un SO.: El shell, los administradores del SO., el Kernel o núcleo.
Prestaciones y servicios de los SO.

Módulo 2: Procesos

Definición y concepto de proceso.
Estados de un proceso. Diagrama de estados. Ciclo de vida de un proceso. Transiciones de estado.
Las operaciones sobre un proceso: creación, manipulación y muerte de un proceso.
El control de un proceso. Estructuras de control del sistema operativo.
Tipos de procesos: los procesos pesados y livianos, hilos o hebras (Threads).
Implementación de hilos (Threads). La creación y ejecución de los Threads. Estado de los threads. Uso de los hilos. Sistemas operativos "multithreaded": aspectos del diseño e implementación de paquetes de Threads. El Concepto de Fibra (Fiber). principios de multitareas

Módulo 3: Planificación de procesadores

Objetivos. Introducción al problema de la planificación: planificación de monoprocesadores y multiprocesadores.
Niveles de planificación: extra largo plazo, largo plazo, mediano plazo y a corto plazo.
Criterios de planificación de los trabajos y de los procesos: política vs. mecanismo.
Administración y gestión de procesos y procesadores: tipos de planificadores. Algoritmos de planificación del procesador.
Algoritmos NON-PREEMPTIVE (sin reemplazo o apropiativos): FCFS (First-Come First-Served), SPF-Shortest Process First (también llamado SPN-Shortest Process Next). Planificación por prioridad.
Algoritmos PREEMPTIVE (con reemplazo en el uso del procesador), Round Robin o torneo cíclico, Menor tiempo restante (SRT Shortest Remaining Time First). Primero el de mayor tasa de respuesta (HRRN).
Planificación con colas de múltiples niveles y realimentación. Planificación con múltiples colas fijas.
Planificación con múltiples colas dinámicas. Planificación de tres niveles. Evaluación y comparación de algoritmos.
Planificación de múltiples procesadores: granularidad, planificación de múltiples procesos y de hilos.
Ejemplos de “scheduler/dispatcher” de sistemas operativos.
Evaluación de desempeño. Detección de cuellos de botellas en el procesador.

Modulo 4: Sincronización y Comunicación entre Procesos

Conceptos de sincronización y comunicación entre procesos.
Problemas concurrentes. Grafos de precedencia. Condiciones de concurrencia (Bernstein).
Especificaciones concurrentes: Fork y Join, Cobegin y coend.
Relaciones entre procesos concurrentes y sus conflictos. Introducción al problema de la región crítica (RC.). Condición de carrera. Solución de la concurrencia por software y hardware.
Algoritmos de sincronización con espera activa: solución simple, espera ocupada por turnos (alternancia), solución de Peterson, algoritmo de Dekker, algoritmo de Lamport o de la panadería.
Algoritmos sin espera activa: semáforos, monitores.
Mecanismos provistos por el hardware. Cola de espera, Semáforos.
Comunicaciones entre procesos: mensajes, IPC: Inter Process Communication, tipos de sincronizaciones mediante mensajes, modelo productor-consumidor, algunos algoritmos para el modelo productor-consumidor.
Deadlocks (interbloqueo, bloqueo mutuo o abrazo mortal). Condiciones necesarias y suficientes. Tipos de recursos.
Ejemplos de abrazo mortal. Prevención, detección, evasión y recuperación de abrazo mortal.
Métodos de representación: grafos y matrices. Grafo de asignación de recursos. Estrategias para tratar Deadlocks. Conflicto en la comunicación entre procesos

Modulo 5: Administración de Memoria Central

Administración de memoria central (MC). Funciones del administrador de la memoria central. Objetivos de la administración de la MC. Asignación y reasignación de direcciones. Espacio de direcciones lógico y físico.
Técnicas de administración sin swapping: Memoria dedicada (máquina desnuda sin SO.), Asignación contigua simple o monitor residente, asignación particionada simple y variable, paginación pura, segmentación simple, manejo de memoria con buddy system.
Técnicas de administración con swapping (intercambio) o sea memoria virtual: swapping, paginación por demanda o bajo solicitud.
Algoritmos de gestión de memoria virtual, sistemas mixtos: segmentación con paginación por demanda.

Módulo 6: Sistema de Gestión de Entrada / Salida

Administración de la Entrada / Salida (I/O scheduler). Funciones del administrador de E/S. Módulos de E/S y la estructura del módulo de E/S.
Función del módulo. Estructura del módulo de E/S.
Las operaciones del hardware de E/S: operación asincrónica, diferencias de velocidades. Los dispositivos y sus interfaces (el hardware de E/S): dispositivos de E/S. Controlador, adaptador o interface de E/S, procesadores de E/S (IOP), dispositivos externos, almacenamiento intermedio de E/S (Buffering), dispositivos internos.
Técnicas de E/S: E/S programada, E/S por interrupciones, E/S por DMA (Acceso Directo a Memoria).
Principios del software de E/S. Metas del software de E/S. Manejadores de interrupciones (Interrupt handler). Drivers de dispositivos. Pasos y controles en una operación de E/S . Software de E/S independiente del dispositivo.
Software de E/S del espacio del usuario. Software de entrada. Software de salida. Procesadores de E/S y Canales de E/S

Módulo 7: Sistema de archivos y sus directorios

Introducción sistema de gestión de archivos (File System).
Concepto de archivo. Tipos de archivos. Atributos de los archivos.
Sistemas basados en cinta y en disco.
Objetivos y funciones del sistema de gestión de archivos. Conflictos. Sistema básico de archivos. La estructura de la información. Archivos mapeados a memoria. Nombres de archivos. La estructura de un archivo. Estructura interna. Descriptores de archivos.
Operaciones sobre archivos: apertura y cierre, creación, escritura, lectura, rebobinado y borrado.
Catalogación de los archivos en el soporte: Área de datos fijos, área de catálogo y área de datos. Administración del espacio de almacenamiento: espacio libre, métodos de asignación. Sistemas de directorio: directorio de dispositivos. Operaciones sobre directorios. Estructuras de directorio.
Métodos de acceso: acceso secuencial, acceso directo, otros métodos de acceso. Métodos de implementación del sistema de archivos. Algoritmos para la administración de archivos.
Protección de archivos: nombre, contraseñas, control de acceso.

Módulo 8: Protección y seguridad

Concepto de seguridad y protección. Concepto de política y mecanismo. Política de seguridad. Principios de las políticas de seguridad. Categorías básicas de las políticas de seguridad. Objetivos de la seguridad y la protección de un sistema. Justificación de la seguridad y protección. Niveles de seguridad en informática. Amenazas a la seguridad. Diseño: principio de los mecanismos. Tipos de seguridad. Supervisión y vigilancia. Supervisión de riesgos de seguridad por el SO.
Seguridad a través del sistema operativo. Funciones de los sistemas de protección en el sistema operativo. Seguridad en el kernel.
Dominios de protección: matriz de accesos. Implementación de la matriz de accesos. Cambio de dominio – switch.
Cambio de contenido de la matriz de accesos. Revocación de permisos. Sistemas basados en capacidades. Seguridad multinivel, autenticación del usuario: validación. Los problemas de la identidad: sus puntos débiles. Amenazas relacionadas con los programas: caballo de troya, puerta trasera, bomba lógica, desbordamiento de pila y de buffer, virus, gusanos, vulnerabilidad. Política de seguridad.
Seguridad para los datos. Seguridad de datos en bases de datos. Métodos de ocultamiento de los datos. Algunos problemas en CRIPTOGRAFÍA.
Seguridad en telecomunicaciones o redes de computadoras. Distribución de llaves. Normas y procedimientos en un sistema de seguridad: estrategia de seguridad, plan de contingencia. Auditorias. Mecanismos y políticas de seguridad en sistemas.

Módulo 9: Sistemas distribuidos

Conceptos de sistemas cliente/servidor y sus variantes.
Conceptos de procesamiento distribuido.
Conceptos de sistemas de archivos en sistemas distribuidos.
Conceptos de control de concurrencia en sistemas distribuidos.
Conceptos de memoria compartida distribuida.
Conceptos sobre transacciones distribuidas

Modulo 10: Sistemas de alto rendimiento

Conceptos de procesadores de alta performance.
Conceptos de procesamiento paralelo.
Conceptos de arquitecturas multiprocesadores.
Generación y ajuste de un sistema operativo. Mediciones del sistema y performance.

BIBLIOGRAFÍA RECOMENDADA PARA EL CURSO (EN INGLÉS) 2014

OBRA: Operating Systems Internals and Design Principles (7th Edition)
AUTOR: Stallings, William
EDITORIAL: Prentice Hall
FECHA: 2011

OBRA: Operating Systems Concepts (9th edition)
AUTOR: Silberschatz, J.L. and Galvin P. B.
EDITORIAL: Addison Wesley
FECHA: 2012

BIBLIOGRAFÍA RECOMENDADA PARA CONSULTA (EN INGLÉS)

OBRA: UNIX Internals - A Practical Approach
AUTOR: Steve D Pate
EDITORIAL: Addison Wesley
FECHA: 1996

OBRA: Advanced programming the UNIX environment
AUTOR: Richard Stevens
EDITORIAL: Addison Wesley
FECHA: 2001

OBRA: UNIX network programming Volume 1
AUTOR: Richard Stevens
EDITORIAL: Prentice Hall
FECHA: 1998

BIBLIOGRAFÍA RECOMENDADA PARA CONSULTA (EN CASTELLANO)

OBRA: Notas sobre Sistemas Operativos - Manual del Alumno - 2 tomos
AUTOR: La Cátedra
EDITORIAL: Ghia
FECHA: 2006
OBSER.: Libro de referencia para el seguimiento de las clases

OBRA: Apuntes de Sistemas Operativos Distribuidos
AUTOR: La Cátedra
EDITORIAL: Ghia
FECHA: 2007
OBSER.: Libro de referencia para el seguimiento de las clases

OBRA: Sistemas Operativos Principios de diseño (Desde la Fifth Edition)
AUTOR: Stallings, William
EDITORIAL: Prentice Hall
FECHA: 2006

OBRA: Sistemas Distribuidos – Conceptos y Diseño (Desde la 3° Edition)
AUTOR: George Coulouris / Jean Dollimore / Tim Kindberg
EDITORIAL: Addison Wesley
FECHA: 2001

OBRA: Sistemas Operativos (Desde la 7ta. edición)
AUTOR: Silberschatz, Galvin & Gagne
EDITORIAL: Mc Graw Hill
FECHA: 2006

METODOLOGÍA DE ENSEÑANZA.

El dictado del curso será del tipo explicativo, participativo e informativo, basado en la discusión de los tópicos desarrollados en el transcurso de las diferentes clases mediante su tratamiento teórico y de ejemplos de aplicaciones prácticas**.**

La introducción de un tema, generalmente es precedida por un diálogo dirigido, con preguntas orientadas hacia el tema a tratar, lo que induce a la participación de todo el grupo.

A partir de esto se desarrolla la exposición teórica con ejercitación práctica en el aula, esta exposición puede ser apoyada por una lectura previa recomendada a los alumnos. Los conceptos impartidos son reforzados y puestos en práctica con los ejercicios propuestos en la Guía de Ejercicios confeccionada por la cátedra. Esa ejercitación permite al alumno confrontar los nuevos conocimientos con los previamente adquiridos y aplicar los conceptos vistos teóricamente, a nuevas situaciones. Algunos ejercicios son presentados, discutidos y resueltos en el aula por el docente

Además la cátedra dispone de una guía de trabajos prácticos para ser desarrollados por los alumnos en forma grupal en el laboratorio de Sistemas Abierto que dispone la cátedra. Cada tema propuesto se verifica en tiempo y forma en el laboratorio observando su correcto funcionamiento en la computadora. Dentro de este ámbito el alumno dispone de atención permanente de docentes para aclarar todas sus consultas. Este estilo de trabajo es abordado durante todas las clases. Cada Trabajo práctico Grupal se deberá defender en forma individual para su aprobación.

Se utilizará material audiovisual cuando las circunstancias así lo requieran.

6.- DESCRIPCIÓN DE LA ACTIVIDAD CURRICULAR

Por parte del Profesor:

- Desarrollo de clases de exposición de temas teóricos Utilizando Pizarrón y en ocasiones con presentaciones por computadora.
- Desarrollo de clases prácticas de resolución de ejercicios de aplicación de los conceptos teóricos.
- Desarrollo de clases prácticas en laboratorio, mostrando ejemplos significativos de la teoría y demostraciones prácticas que deben realizar los alumnos.
- Actualización de contenidos en la página de la cátedra donde se encuentra toda la documentación de la asignatura y medio de comunicación para envío de noticias, material y dar respuesta a los requerimientos de los alumnos (acceso vía e-mail de y a los alumnos).

Por parte de los alumnos:

- Resolución individual de ejercicios de aplicación de la teoría propuestos por el profesor.
- Desarrollo e implementación grupal de una a serie de trabajos Prácticos diseñados especialmente para que el grupo de alumnos los programe.

Material Didáctico:

- Diapositivas Power Point de Clase sobre temas teóricos
- Guías de Trabajos Prácticos y Ejemplos de Resolución de Ejercicios desarrollados por la cátedra
- Procedimientos escritos para diversos procesos.
- Bibliografía básica y avanzada.
- Uso del sitio de la Asignatura.

Envío y atención de mail para consultas

7.- EXPERIENCIAS DE LABORATORIO, TALLER O TRABAJOS DE CAMPO

NOTA: Todos los trabajos prácticos que se realicen en ésta materia serán corregidos en un entorno GNU/Linux, utilizando UBUNTU. Para mayor detalle de las versiones utilizadas concurrir al laboratorio 266.
En caso de que el alumnado decida realizar los trabajos en otra plataforma (dentro del mundo Unix), o en otra distribución, deberá tomar las precauciones necesarias para que el producto entregado pueda ser ejecutado en dicho ambiente. En caso de que el producto entregado no cumpla con éstas indicaciones, el trabajo práctico será considerado como no entregado.

Trabajo Práctico Nro. 1:

Programación de scripts básicos, con awk y sed sistema operativo GNU/Linux (Grupal)

Trabajo Práctico Nro. 2:

Programación en ANSI-C, o C++, bajo sistema operativo GNU/Linux, para resolver ejercicios de concurrencia, manejo de hijos, señales y FIFOS (Grupal)

Trabajo Práctico Nro. 3:

Programación en ANSI-C, o C++, bajo sistema operativo GNU/Linux, para resolver ejercicios de manejo de semáforos, memoria compartida, Threads y Sockets (Grupal)

Trabajo Práctico Nro. 4:

Programación en ANSI-C, o C++, bajo sistema operativo GNU/Linux, de un ejercicio integrador basado en el desarrollo de un juego para consola gráfica (Grupal)

Defensa de TP's (Coloquio) y Recuperatorios

Todos los Trabajos prácticos serán probados por los docentes y defendidos por los alumnos en el Laboratorio. En las defensas podrán estar presentes además de los docentes del curso, el jefe de cátedra, y el jefe de trabajos prácticos.

8.- USO DE COMPUTADORAS

En la asignatura se utilizan profusamente computadoras en experiencias de simulación y de programación de Sistemas como complemento práctico de la Teoría. Para ello se recurrirá al Laboratorio específico de Sistemas Operativos (aula 266).

9.- METODOLOGÍA DE EVALUACIÓN

- Esta asignaturas se evaluará de acuerdo a la reglamentación vigente en la Universidad y la que se detalla en el "Reglamento de cursado y aprobación de la materia".
- Se efectuarán dos evaluaciones parciales: El primero al promediar el dictado del curso y el segundo al finalizar el mismo según el Calendario Académico.
- Asimismo y como **condición necesaria para la aprobación del curso** se examinará al alumnado mediante una Guía de Trabajos Prácticos que se desarrollará durante el transcurso del mismo, además de las exposiciones orales que efectuarán los alumnos sobre los T.P., cuestionarios o problemas teóricos planteados.
- Además se requiere una asistencia a clase no inferior al 75%, se hace un seguimiento del trabajo realizado por cada integrante en cada clase.

El conjunto formado por los Trabajos Prácticos y las evaluaciones parciales serán el instrumento para medir el rendimiento y la aprobación de la cursada.

Aprobación y cursada

A los fines de la aprobación de la materia, se considera “la última nota obtenida” en cada uno de los exámenes rendidos (en primera instancia ó recuperatorios).

a) Por régimen de promoción, sin examen final, se considera la materia **aprobada**, cuando la calificación es igual o superior a 7 (siete) a través de exámenes parciales y recuperatorios, en las fechas indicadas en el cronograma.
b) Si el alumno no llena los requisitos para promover (calificación superior o igual a 4 pero inferior a 7 puntos), queda en condición de **cursada**. Para su aprobación definitiva tiene que rendir posteriormente un examen final. La validez de la cursada será de 5 turnos consecutivos de examen final. Dichos turnos serán contados a partir del turno inmediato siguiente al periodo de cursada.
c) El alumno que tenga 1 (un) aplazo en las evaluaciones y/o recuperatorios, y haya estado presente en todas las instancias evaluativas, pierde la materia y se considera **desaprobado.**
d) Aquel alumno que tenga al menos 1 (un) examen cuya evaluación final sea ausente (considerando parcial y recuperatorio), se considera **ausente.**

Régimen de Trabajos Prácticos

1- En fecha de entrega del TP, se hará una corrección grupal de ejercicios en forma arbitraria para cada grupo.
2- La NO presentación del TP o estar incompleto en la fecha propuesta significa su desaprobación en primera instancia (es para todos los integrantes del grupo).
3- La realización de cada TP será grupal pero su evaluación individual a través de un examen escrito en los trabajos prácticos 1 a 3 y oral para el trabajo práctico 4 .
4- El alumno que no apruebe la evaluación del TP o NO se entregó en fecha establecida, tendrá una nueva fecha que es a la semana siguiente del establecido inicialmente en el cronograma.
5- El alumno que desapruebe 2(dos) TP en segundas instancias (por ausencia, estar incorrecto, y /o incompleto o no responder correctamente el coloquio) desaprobará la materia.

Examen Final

- Los alumnos pueden rendir examen final bajo dos modalidades **regular** o **libre.**
- Para rendir examen como **regular** deberá tener la materia cursada y no haberse operado el vencimiento de la misma.
- Deberán rendir como regular los que obtengan entre cuatro y seis en los parciales o sus recuperatorios.
- Para rendir examen como **libre** tendrán que ajustarse a la reglamentación vigente.
- La mesa examinadora considerará válidas las inscripciones que consten en las actas proporcionadas por la oficina de alumnos.
- Cada alumno rendirá el final con el programa vigente.

Condiciones para rendir EXAMEN LIBRE

Por Resolución Nº 142 del H.C.S., se autoriza a rendir "**exámenes libres**" de todas las asignaturas, a los alumnos de las tres Carreras pertenecientes al Departamento de Ingeniería e Investigaciones Tecnológicas.

Todos los alumnos estarán en condiciones de rendir exámenes libres, siempre y cuando hayan aprobado las materias correlativas correspondientes.

Dicha instancia examinadora se deberá llevar a cabo en una de las fechas de convocatoria a exámenes finales, y este Departamento ha dispuesto que ello se realice, únicamente, en la **primera fecha de cada turno**.

Para mayor detalle sobre la forma de rendir exámenes libres y los requerimientos a cumplir antes de presentarse en la llamada correspondiente ver la sección REGLAMENTO DE CURSADA LIBRE DE LA CÁTEDRA SISTEMAS DE COMPUTACIÓN II (PLAN 1997) / SISTEMAS OPERATIVOS (PLAN 2009) más adelante en el presente documento

CALENDARIO DE ACTIVIDADES (modalidad presencial)

PLANIFICACIÓN DOCENTE PARA EL AÑO 2014

DURACIÓN DE CADA CURSO:

- Teórica: Aprox. 8 clases de 4 horas. (32 horas)
- Práctica: Aprox. 9 clases de 4 horas. (36 horas)
- Laboratorio: Aprox. 13 clases de 4 horas. (52 horas)

HORARIO: Según el fijado para cada curso (turno mañana 8 a 12 / turno noche 19 a 23)

CRONOGRAMA DE ACTIVIDADES DE LA PLANIFICACIÓN POR CURSO

Clase 1: Introductoria-Práctica. Presupuesto de tiempo: 4 Hs.

Fecha: Comienzo del ciclo lectivo 2014- (25/08/2014)

Objetivos:

- Definir la metodología para el futuro desarrollo del curso y dar los lineamientos introductorios al curso. Explicar la metodología de evaluación de TPs y exámenes parciales
- Introducción al sistema operativo GNU/Linux

Tipo de conocimiento:

- Práctico

Evaluación del Módulo:

- Primer parcial

MÓDULO 1: Presupuesto de tiempo: 2 Hs.

Objetivos:

- Que el alumno incorpore un enfoque introductorio sobre los Sistemas Operativos, sus interfaces, los servicios que brinda, su funcionamiento y conozca la terminología básica y conceptual de los S.O. y sus ambientes de trabajo.

Tipo de conocimiento:

- Teórico

Evaluación del Módulo:

- Primer parcial

MÓDULO 2: Presupuesto de tiempo: 2 Hs.

Objetivos:

- Que el alumno adquiera los conocimientos sobre las distintas modalidades de procesamiento e incorpore los conceptos fundamentales sobre organización del ambiente de ejecución.

Tipo de conocimiento:

- Teórico

Evaluación del Módulo:

- Primer parcial

PRÁCTICO 1: Presupuesto de tiempo: 8 Hs.

Objetivos:

- Que el alumno incorpore los conocimientos para la codificación de scripts básicos y con las herramientas awk y sed

Tipo de conocimiento:

- Práctica de laboratorio

Evaluación del Módulo:

- Coloquio del TP

EVALUACIÓN PRÁCTICO 1: Presupuesto de tiempo: 1 Hs.

Objetivos:

- Evaluar a los alumnos sobre los conocimientos de GNU/Linux adquiridos durante el trabajo práctico 1

MÓDULO 3: Presupuesto de tiempo: 8 Hs.

Objetivos:

- Que el alumno se familiarice con los conceptos y los medios de la planificación del procesador y de los procesos, en especial en el largo, mediano y corto plazo.

Tipo de conocimiento:

- Teórico y práctico

Evaluación del Módulo:

- Primer parcial

MÓDULO 4: Presupuesto de tiempo: 10 Hs.

Objetivos:

- Que el alumno integre los conceptos fundamentales sobre los recursos compartidos, sincronización y comunicación entre procesos.

Tipo de conocimiento:

- Teórico y práctico

Evaluación del Módulo:

- Primer parcial

PRÁCTICO 2: Presupuesto de tiempo: 6 Hs.

Objetivos:

- Que el alumno incorpore los conocimientos para la codificación en “C” de procesos concurrentes

Tipo de conocimiento:

- Práctica de laboratorio

Evaluación del Módulo:

- Coloquio del TP

EVALUACIÓN PRÁCTICO 2: Presupuesto de tiempo: 1 Hs.

Objetivos:

- Evaluar a los alumnos sobre los conocimientos de GNU/Linux adquiridos durante el trabajo práctico 2

EVALUACIÓN 1: Presupuesto de tiempo: 4 Hs.

Objetivos:

- Evaluar a los alumnos sobre los conocimientos teóricos y prácticos.

MÓDULO 5: Presupuesto de tiempo: 10 Hs.

Objetivos:

- Que el alumno concrete los conceptos sobre la administración de la Memoria Central, en especial las particiones y los conceptos de asignación, paginación y segmentación.

Tipo de conocimiento:

- Teórico y práctico

Evaluación del Módulo:

- Segundo parcial

MÓDULO 6: Presupuesto de tiempo: 6 Hs.

Objetivos:

- Que el alumno incorpore los conceptos sobre la administración de los dispositivos de Entrada - Salida.

Tipo de conocimiento:

- Teórico y práctico

Evaluación del Módulo:

- Segundo parcial

MÓDULO 7: Presupuesto de tiempo: 6 Hs.

Objetivos:

- Que el alumno conozca los métodos de acceso para el almacenamiento y la recuperación de la información en los soportes como también la administración de la misma.

Tipo de conocimiento:

- Teórico y práctico

Evaluación del Módulo:

- Tercer parcial

MÓDULO 8: Presupuesto de tiempo: 1 Hs.

Objetivos:

- Que el alumno conozca los fundamentos y los conceptos sobre el manejo de la protección y la seguridad de un centro de cómputo y el S.O.

Tipo de conocimiento:

- Teórico

Evaluación del Módulo:

- Segundo parcial

PRÁCTICO 3: Presupuesto de tiempo: 10 Hs.

Objetivos:

- Que el alumno incorpore los conocimientos para la codificación en "C" de threads y las herramientas de sincronización y comunicación de procesos

Tipo de conocimiento:

- Práctica de laboratorio

Evaluación del Módulo:

- Coloquio del TP

EVALUACIÓN PRÁCTICO 3: Presupuesto de tiempo: 1 Hs.

Objetivos:

- Evaluar a los alumnos sobre los conocimientos de GNU/Linux adquiridos durante el trabajo práctico 3

EVALUACIÓN 2: Presupuesto de tiempo: 4 Hs.

Objetivos:

- Evaluar a los alumnos sobre los conocimientos teóricos y prácticos.

MÓDULO 9: Presupuesto de tiempo: 1 Hs.

Objetivos:

- Que el alumno adquiera los conceptos básicos sobre métrica de sistemas

Tipo de conocimiento:

- Teórico

Evaluación del Módulo:

- Sin evaluación

MÓDULO 10: Presupuesto de tiempo: 6 Hs.

Objetivos:

- Que el alumno adquiera los conceptos básicos sobre los sistemas operativos distribuidos, sus problemáticas y la forma de implementar las soluciones

Tipo de conocimiento:

- Teórico

Evaluación del Módulo:

- Sin evaluación

PRÁCTICO 4: Presupuesto de tiempo: 10 Hs.

Objetivos:

- Que el alumno consolide todos los conocimientos incorporados en el cuatrimestre realizando un trabajo práctico integrador

Tipo de conocimiento:

- Práctica de laboratorio

Evaluación del Módulo:

- Coloquio del TP

RECUPERACIÓN 1: Presupuesto de tiempo: 4 Hs.

Objetivos:

- Que los alumnos tengan la posibilidad de recuperar los exámenes que tengan aplazados o con notas menores a 7

CALENDARIO DE ACTIVIDADES (modalidad semi-presencial)

PLANIFICACIÓN DOCENTE PARA EL AÑO 2014

DURACIÓN DE CADA CURSO:

- Actividad presencial: 16 clases de 4 horas. (64 horas) un encuentro cada 7 días

HORARIO: viernes c/7 días turno noche de 19 a 23

CRONOGRAMA DE ACTIVIDADES:

Fecha	Presencial	Actividad/es
viernes, agosto 29, 2014	Si	✓ Presentación de la modalidad de cursada ✓ Módulo 1 / 2 ✓ Linux - Scripts básicos
viernes, septiembre 5, 2014	Si	✓ Módulo 3 teoría y práctica ✓ Linux - Scripts avanzados
viernes, septiembre 12, 2014	Si	✓ Módulo 4 ✓ Linux – Make, Fork y Wait
viernes, septiembre 19, 2014	Si	✓ Práctica Módulo 4 ✓ Linux – Señales, Exec y popen ➢ Entrega del TP1
viernes, septiembre 26, 2014	Si	✓ Consultas P1 ✓ Defensa TP1 ➢ Entrega del TP1 fecha tardía
viernes, octubre 3, 2014	Si	➢ Primer parcial (1-4)
viernes, octubre 10, 2014	Si	✓ Módulo 5 ✓ Linux – Mc, Sem, Threads y Sockets ➢ Entrega del TP2
viernes, octubre 17, 2014	Si	✓ Práctica Módulo 5 ✓ Defensa TP2
viernes, octubre 24, 2014	Si	✓ Módulo 6 y 7 ✓ Linux – SDL
viernes, octubre 31, 2014	Si	✓ Raid ✓ Práctica Módulo 6 y 7 ➢ Entrega del TP3
viernes, noviembre 7, 2014	Si	✓ Consultas P2 ✓ Control avances TP4 ✓ Defensa TP3 ➢ Entrega del TP3 fecha tardía
viernes, noviembre 14, 2014	Si	➢ Parcial 2 (5-8)
viernes, noviembre 21, 2014	Si	✓ Consultas Recuperatorio ➢ Entrega del TP4
viernes, noviembre 28, 2014	Si	➢ Recuperatorio
viernes, diciembre 5, 2014	Si	➢ Reentrega del TP4 (sólo grupos citados)
viernes, diciembre 12, 2014	Si	➢ Reentrega del TP4 (sólo grupos citados) ➢ Cierre de actas

REGLAMENTO DE PROMOCIÓN y NORMAS DE LA CÁTEDRA (Cursada normal)

1. La aprobación de la asignatura *Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)* en el período lectivo **2014** se basará en:
   1.1. La Normativa vigente en la Universidad,
   1.2. Las Normas Básicas de la Cátedra *Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)* que se detallan a continuación en tanto reglen aspectos no normados por los elementos anteriores.
   1.3. El Reglamento Interno de la *Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)* que se detalla a continuación, en este documento.

NORMAS BÁSICAS DE LA CÁTEDRA

1. El dictado de la materia se dividirá en aproximadamente 32 clases teóricas y clases prácticas, según calendario adjunto, las que incluirán clases teóricas, práctica, de evaluación y de recuperación.

2. En las clases de contenido teórico se desarrollarán los temas teóricos establecidos en el programa analítico adjunto. En las clases prácticas los alumnos, orientados por los docentes a cargo de las mismas, resolverán problemas y ejercicios de aplicación de los temas vistos en clase y los de la presente guía de trabajos prácticos.

3. La aprobación de los trabajos prácticos (Firma de Libreta), se obtendrá a través de:
   3.1. La presentación y aprobación de los trabajos prácticos, según lo detallado en la presente.
   3.2. La aprobación de, al menos, dos exámenes parciales en las fechas y condiciones establecidas en el calendario adjunto y en el régimen de cursado y aprobación de ambas asignaturas siguiente:

Régimen de cursado y aprobación de la asignatura *Sistemas de Computación II (Plan 1997) / Sistemas Operativos* (Plan 2009)

1. ASISTENCIAS:

- Se requiere una asistencia a clases no inferior al 75% (setenta y cinco %) tanto para la modalidad presencia como la semipresencial. El incumplimiento de este requisito coloca al alumno en condición de "ausente" o "desaprobado".

2.- RÉGIMEN DE PROMOCIÓN POR EXÁMENES PARCIALES Y RECUPERATORIOS:

- La asignatura se aprueba por régimen de promoción por exámenes parciales y/o recuperatorios.
- Para esta asignatura el cursado es cuatrimestral y habrá 2 (dos) evaluaciones parciales: uno por bimestre. Las instancias recuperatorias serán 1 (una). En esta asignatura se entiende **"ausente"** al alumno que no posea 2 (dos) evaluación parciales rendidas y no haya entregado más de 2 (dos) trabajos prácticos.
- Los exámenes parciales (y sus recuperatorios) se entenderán **"aprobados"** cuando la calificación asignada, en una escala de 0 a 10 puntos, resulte superior o igual a 7 (siete) puntos.
- La asignatura se entenderá **"aprobada"** cuando se aprueben todos los exámenes parciales (en primera instancia o por recuperatorio). La calificación asignada al examen recuperatorio (cualquiera sea el resultado), **anula y reemplaza**, a todos los efectos, la obtenida en el examen parcial que se recupera. La calificación final se calculará como promedio de los exámenes parciales o el último recuperatorio de cada parcial, rendidos y aprobados. Es importante mencionar que en caso de tener una nota entre 4 (cauto) y 6 (seis) en el parcial y sacarse un 2 (dos) en el recuperatorio, la materia se considerará como **"desaprobada"**
- De esta manera la calificación final necesaria para que la asignatura resulte **"aprobada"** deberá ser superior o igual a 7 (siete) puntos, pero no se podrán poseer notas menores a 7 (siete).

3. RÉGIMEN NO PROMOCIONADO DE PARCIALES Y SUS RECUPERATORIOS:

- En la asignatura, los exámenes parciales (y sus recuperatorios) calificados con 3 (tres) o menos puntos se entenderán **"aplazados"** y podrán ser recuperados.
- Si el alumno al finalizar la cursada tiene algún parcial (o recuperatorio) y/o trabajo práctico calificados con aplazo, se considerará **"aplazada"** la materia y deberá ser recursada en otro cuatrimestre.
- Los exámenes parciales calificados con 4 (cuatro), 5 (cinco) ó 6 (seis) puntos, se entenderán **"aprobados"** y podrán ser recuperados (en caso de que el alumno desee la promoción de la materia), pero no se podrá con estas notas conseguir la calificación final de "**Aprobado**", teniendo en este caso la condición final de **"Cursada"**.
- La asignatura con calificación final, calculada como promedio de los exámenes parciales (o los recuperatorios correspondientes) rendidos y no aplazados, de 4 (cuatro), 5 (cinco) o 6 (seis) puntos, se entenderán **"cursada"** y podrá ser aprobada a través de un "**examen final**".
- La calificación necesaria para aprobar el "**examen final**" será de 4 o más puntos

4.- LA VALIDEZ DE LA ASIGNATURA "CURSADA"

- La validez de la asignatura **"cursada"** se rige bajo las normas de la facultad, por lo que se deberá consultar con las autoridades pertinentes.

5.- PREREQUISITO CONDICIONANTE PARA RENDIR LOS EXAMENES PARCIALES:

- Esta asignatura requiere reglamentariamente la aprobación de Trabajos Prácticos, entonces, dicha aprobación se entiende como requisito previo para rendir los exámenes parciales y los recuperatorios correspondientes.

REGLAMENTO INTERNO DE LA CÁTEDRA SISTEMAS DE COMPUTACIÓN II (PLAN 1997) / SISTEMAS OPERATIVOS (PLAN 2009)

1. OBJETIVO:

- Dar las bases normativas por las que se regirá el funcionamiento y el desarrollo operativo de la cátedra.

2. ALCANCES:

- El presente Reglamento **NO EXCLUYE** a la reglamentación vigente, sino todo lo contrario, pretende complementarla para lograr las metas operativas propuestas para cada curso en particular.

3. CONTENIDO:

a) **DE LOS PROGRAMAS**: El contenido es el indicado en el programa analítico de la materia.

b) **DEL CRONOGRAMA DE ACTIVIDADES**: Se ajustará de acuerdo al presupuesto de tiempo previsto en la planificación docente y se formalizará el primer día de clase en cada curso en particular.

c) **DE LA ASISTENCIA**: Es de recalcar que la asistencia, en el caso específico de ésta materia, juega un rol importante debido al intenso ritmo que se impartirá al dictado de las clases teóricas, por lo tanto se recomienda al alumno concurrir a dichas clases, siendo de su exclusiva responsabilidad cumplir con este requisito. Pero se aplicará el punto 1 del Régimen de cursado y aprobación de las asignaturas *Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)*.

d) **DEL HORARIO**: En el inicio de la clase, la puntualidad es importante a los fines de constituir un ambiente ordenado. Se recomienda al alumnado el cumplimiento de este requerimiento. En particular también se recomienda la permanencia dentro del aula mientras se desarrollan las clases.

e) **DE LAS CLASES TEÓRICAS**: El Docente y sus Ayudantes dictarán la materia tratando de seguir la secuencia estricta de los módulos y la Planificación propuesta. El desarrollo tendrá un carácter ampliamente comunicativo que permita la participación del alumnado. El método a aplicar será explicativo-inductivo-deductivo. Tanto el docente o sus Ayudantes evacuarán las dudas que surjan durante el dictado de las clases o de los T.P.

f) **DE LOS TRABAJOS PRÁCTICOS**: Los alumnos confeccionarán una serie de Trabajos Prácticos (TPs.), para ello se dispondrá de una Guía de T.P. (adjunta al presente documento).
Cada Guía deberá ser completada en la fecha establecida por la Cátedra o el docente a cargo del curso. La totalidad de las guías formarán una **"carpeta de T.P."**
Los T.P. se dividirán en dos categorías: 1) Optativos y 2) Obligatorios.
Cada guía deberá ser entregada, por el alumno o el grupo de alumnos, **en la fecha planificada a los efectos de ser corregida**.
OBSERVACIÓN: LOS T.P. NO ENTREGADOS EN FECHA SE CONSIDERAN NO APROBADOS.
La guía corregida por la cátedra será devuelta con las observaciones correspondientes para que los alumnos procedan a rectificar lo solicitado. Una vez cumplimentado por los alumnos, en el plazo fijado, los T.P. serán entregados a la cátedra para su aprobación. La cátedra firmará la aprobación parcial de cada Guía y devolverá el original para que cada alumno pueda disponer de una constancia de la aprobación, la que integrará una "carpeta de TPs". Todos los TPs originales aprobados formarán una "carpeta de T.P. originales" que deberá ser presentada al final de la cursada.

g) **DE LA PRESENTACIÓN DE LOS T.P.:** La presentación se deberá realizar en dos soportes: Hojas de papel y medio electrónico (ver reglamento de entrega más adelante en esta guía).
La presentación que se realiza en hojas de papel, deberá ser normalizada en papel A-4, o carta (no se aceptarán entregas en papel oficio), y los contenidos impresos se ajustarán a las **"Normas para la presentación escrita de los Trabajos Prácticos"** que figuran en la guía de T.P.

h) **DE LA EVALUACIÓN DE LOS T.P.**: Todos los puntos se evaluarán mediante las consideraciones en particular de cada ítem siguiente:

- Desarrollo por temas (extensión).
- Contenidos (Calidad y en el caso de programas: funcionamiento).
- Criterios.
- Síntesis.
- Definiciones (acotaciones).
- Alcances.
- Investigación Bibliográfica.
- Presentación.

El conjunto de notas dará como resultado la aprobación o desaprobación del T.P. en particular.

i) **DE LA REGULARIZACIÓN DE LA MATERIA:** Para la firma de la Libreta, el alumno deberá presentar:
   - La Libreta Universitaria.
   - Haber aprobado los T.P. realizados durante el curso ya sean grupales o individuales.
   - Tener todos los parciales aprobados y cumplir con lo dispuesto en el Régimen de cursado y aprobación de la asignatura Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009).
   - Ser alumno regular.
   - Realizar el "POSTEST" que propondrá la Cátedra y la encuesta.

j) **DE LAS EVALUACIONES DURANTE EL CURSO:** Habrá dos evaluaciones parciales durante el curso. El docente fijará con cada curso fecha de cada uno de esos parciales y la del recuperatorio.
Habrá un recuperatorios en el que podrá rendirse uno de los dos parciales según lo especificado en el régimen de aprobación de la materia.
Los T.P. grupales serán expuestos en el pizarrón o en una reunión grupal con el Jefe de Trabajos Prácticos o docente del curso, por cada integrante del grupo a los fines de examinar su participación en el desarrollo del T.P. y que dará lugar a una evaluación de cada presentación individual. Además de considerar una nota única por cada T.P. grupal.

k) **DE LAS EVALUACIONES FINALES:** Los mismos pueden ser Teórico-prácticos y en forma escrita y/u oral, según lo aconsejen las circunstancias.
La examinación se hará a través de un Tribunal Examinador. Para poder rendir el examen final los alumnos deberán tener regularizada la materia y la correlatividades respectivas de esta materia.

REGLAMENTO DE CURSADA LIBRE DE LA CÁTEDRA SISTEMAS DE COMPUTACIÓN II (PLAN 1997) / SISTEMAS OPERATIVOS (PLAN 2009)

1. OBJETIVO:

- Dar las bases normativas por las que se implementará la aprobación de la materia a través de exámenes libres para conseguir la aprobación de la cátedra.

2. ALCANCES:

- El presente Reglamento **NO EXCLUYE** a la reglamentación vigente, sino todo lo contrario, pretende complementarla para lograr las metas operativas propuestas para cada curso en particular.

3. CONTENIDO DEL EXAMEN LIBRE:

a) **DE LOS TRABAJOS PRÁCTICOS:** El alumno que desee rendir la materia en condición de libre deberá efectuar **TODOS** los trabajos prácticos que la cátedra haya dispuesto para el cuatrimestre en curso vigente en la guía de trabajos prácticos que suministra la materia, confeccionándolos y teniéndolos que presentar con 15 días de anticipación a la fecha de rendir el examen libre. La vigencia de los trabajos prácticos para el examen libre será desde el comienzo del cuatrimestre correspondiente al que se quiere rendir el examen libre hasta el comienzo del próximo cuatrimestre. Esto quiere decir que por ejemplo si se desea rendir el examen libre correspondiente a la cursada del primer cuatrimestre 2014, la llamada a examen en la que el alumno podrá presentarse son las correspondientes a julio de 2014, en caso de querer rendir el examen libre correspondiente a la cursada del segundo cuatrimestre de 2014, las llamadas en las que podrá presentarse son las de diciembre de 2014, y marzo de 2015 (siempre en la primera fecha del llamado salvo que el jefe de cátedra informe lo contrario).

b) **DE LA EVALUACIÓN DE LOS T.P.**: Los trabajos prácticos entregados por el alumno que rinde el examen libre serán evaluados en los 15 días que hay hasta la fecha del examen final por los docentes de la cátedra y en caso de estar bien, el alumno deberá rendir un coloquio como primera parte del examen final. Los puntos se evaluarán mediante las consideraciones en particular de cada ítem siguiente:

- Desarrollo por temas (extensión).
- Contenidos (Calidad y en el caso de programas: funcionamiento).
- Criterios.
- Síntesis.
- Creatividad.
- Definiciones (acotaciones).
- Alcances.
- Investigación Bibliográfica.
- Presentación.

El conjunto de notas dará como resultado la aprobación o desaprobación de los trabajos prácticos.

c) **DE LA EVALUACIÓN FINAL:** En caso de aprobar los trabajos prácticos (tanto la presentación, como el coloquio), el alumno deberá rendir un examen final para la condición de libre, que tendrá una primera parte práctica escrita (conteniendo ejercicios tanto de la práctica de clases como de la práctica de laboratorio). En caso de aprobar dicho examen deberá pasar un examen teórico con carácter oral que incluirá todo el contenido de la materia que se indica en el PROGRAMA ANÁLITICO de la materia.

"Certifico que el presente programa de estudios de la asignatura *Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)* es el vigente para el segundo cuatrimestre del ciclo lectivo 2014, guarda consistencia con los contenidos mínimos del plan de estudios y se encuentra convenientemente actualizado"

Fabio E. Rivalta
Jefe de Cátedra
Universidad Nacional de La Matanza
25/08/2014

COMENTARIOS PREVIOS

Estos comentarios la opinión de la cátedra respecto de la metodología de estudio, conocimientos previos y la infraestructura necesaria; para el correcto cursado de la materia. Como estos aspectos no dependen exclusivamente del esfuerzo de la cátedra de S.O., es posible que algunas de estas opiniones no sean realidad aun.

RESPECTO DE LOS CONOCIMIENTOS PREVIOS NECESARIOS

Los conocimientos previos que se requieren del alumnado para comprender la temática de la materia y estudiarla con cierto grado de profundidad, básicamente se agrupan en los cuatro siguientes campos: hardware, software, estructuras de datos y algoritmia y dentro de ellos específicamente los siguientes temas:

Hardware:

- Autómatas finitos,
- Álgebra de códigos y álgebra binaria,
- Conceptos de arquitecturas computacionales,
- Direccionamientos del procesador,
- Memoria principal, rango y resolución,
- Periféricos, controladores, canales, interfases, dispositivos,
- Distintos soportes de información, grabación y recuperación,
- Registros de la CPU, ALU, unidad de control,
- Interrupciones.

Software:

- Instrucciones primitivas, lenguajes de máquinas, niveles de lenguajes,
- Instrucciones comunes y privilegiadas, macros
- Concepto de trabajo, paso de trabajo, proceso
- Concepto de traductores y editores de enlace
- Concepto de editores de texto
- Concepto de programa, rutina, autorutina y corutina
- Concepto de declaraciones, declarativas, variables locales y globales.
- Procedimientos, recursividad.

Estructura de datos:

- Registros, archivos, punteros, operadores,
- Arreglos, estructuras, unión, apuntadores o punteros,
- Proposiciones y asignaciones,
- Expresiones,
- Vector, tabla, cola, lista, pila, árboles

Algoritmia:

- Lenguaje c, Pascal o Modula 2
- Lenguajes orientados a objetos (c++)

El alumno que entienda no conocer alguno de los puntos antes citados debería consultar bibliografía adecuada, con el propósito de adquirir el o los conocimientos en cuestión. Este punto es fundamental ya que estos conceptos son esenciales para la comprensión de la materia.

RESPECTO DEL TRABAJO A REALIZAR

- En cuanto a los esfuerzos que el alumno debe dedicar normalmente a esta materia comprenden dos aspectos: uno teórico y otro práctico.
- El teórico consiste en el seguimiento de las clases, que en general se prevé para el cuatrimestre, de cuatro horas reloj por clase con un total de 28 clases para el curso, al cual se deberá agregar un tiempo adicional de lectura de texto (ver bibliografía recomendada), con igual carga horaria. Se recomienda en especial prestar atención a las lecturas de los textos, sobre todo las propuestas en idioma inglés.
- El aspecto práctico queda determinada por la experiencia, con que cuenta cada alumno, en programación mediante lenguajes estructurados y su modalidad de trabajo en laboratorios (cerrados, colaboración con grupos o individual), ya sea hogareña o en los puestos de trabajo cotidiano. De todas formas consideramos que el mínimo esfuerzo en la faz práctica requiere de dos a tres horas diarias de dedicación frente al computador y el mismo tiempo de lectura o estudio por clase.
- Las clases prácticas serán desarrolladas por los Docentes auxiliares en el horario determinado para estas tareas.

RESPECTO DEL LABORATORIO (AULA 266)

- Con respecto al laboratorio, consideramos interesante que los alumnos utilicen el equipamiento ordenadamente solicitando los respectivos turnos con el debido tiempo. Que se dispongan a trabajar no más de tres a cuatro alumnos por máquina.
- El laboratorio deberá estar en condiciones de uso antes y luego de que los alumnos de la cátedra accedieron a la instalación. Respetando la normativa establecida para su correcto uso.
- El trabajo en el laboratorio deberá estar organizado por turnos en el que los alumnos se presentan al mismo, habiendo previamente elaborado la práctica o ejercicios. Recomendamos especialmente no perder tiempo (y hacer perder tiempo), con improvisaciones sin previa fundamentación. Los Ayudantes o instructores estarán disponibles para resolver las consultas específicas que le formulen los alumnos sobre los trabajos y ejercicios propuestos.
- La experiencia nos dice que durante el tiempo previsto, los alumnos no completan las tareas solicitadas de acuerdo al planeamiento propuesto para el desarrollo del curso y solo lo hacen a último momento, por lo que los horarios del laboratorio se satura, entonces también es recomendable que los alumnos utilicen racionalmente al laboratorio dentro de la amplitud horaria que les fuera asignada y no padezcan los efectos de la demanda de último momento.

RECOMENDACIONES PARA DISEÑAR SOFTWARE Y LLEGAR CON ÉXITO A LOS OBJETIVOS.

A los efectos de construir buenos programas y permitir comprender las ideas que él o los programadores desean expresar o comunicar y también facilitar la lectura, sugerimos esta pequeña guía que ha demostrado ser útil en el desarrollo de proyectos en varios ciclos lectivos.

Esta guía se incluye para que los alumnos minimicen la pérdida de tiempo en las tareas grupales y faciliten la evaluación del trabajo por los docentes y permitan realizar correcciones, depuraciones o mantenimientos que generalmente presentan los trabajos prácticos realizados.

Generalmente, las tareas de corrección y depuración son realizadas por otras personas con criterios distintos y modalidades de trabajo también distintas, entonces, para minimizar éstos problemas y sin quitar importancia a la labor creativa necesaria en el diseño de cualquier software, sugerimos una serie de recomendaciones generales, que necesariamente son subjetivas y por tanto discutibles en su validez, pero que han demostrado ser útiles durante la experiencia acumulada en más de 20 años de labor educativa.

Estas recomendaciones, fundamentalmente van dirigidas a alumnos diseñadores de software sin experiencia en proyectos con cierto grado de dificultades y magnitud mediana, por lo que pretendemos crearles un buen hábito desde el inicio y no descuidar los detalles que pudieran ser causa de ineficiencias futuras.

Los aspectos a tener en cuenta son cuatro, a saber, organizativas, de nomenclatura, de identificadores, técnicas y de presentación.

ASPECTO ORGANIZATIVO

Este aspecto es importante a los efectos de que el grupo alumnos aúnen esfuerzos en el diseño y se constituya un real equipo de trabajo con roles definidos y objetivos comprendidos por todos.

Las fases son:

a) Reunión previa: de constitución del grupo de trabajo y toma de conocimiento de la situación de cada integrante y donde se establecen lugares, horarios de reuniones, restricciones, los medios de comunicación, etc. Logrado esto puede continuarse con la siguiente tarea.
b) Comprensión del problema: en el que se analiza globalmente el problema a ser resuelto, generalmente bosquejado en un conjunto de especificaciones o deseos que se entregan inicialmente a los integrantes del grupo.
c) Fase de análisis del problema: en la que se estudia conjuntamente el planteo inicial, los sucesivos objetivos, los eventos que se encadenan, la inicialización, la secuencialidad, el paralelismo, el uso y la desactivación o destrucción de objetos, etc.

En esta fase se deben resolver los siguientes puntos:

1. Eliminar ambigüedades e incoherencias
2. Controlar que las especificaciones iniciales del problema contengan toda la información pertinente sin lagunas y si las hubieran determinar cómo se las cubriría y quien será el responsable de resolverla.
3. Utilizar una estructura (algoritmo) conocida y sencilla con una nomenclatura (codificación) clara y precisa.

Otras consideraciones que deben resolverse desde el punto de vista organizacional en esta fase, son las recomendaciones para el diseño de software, a saber:

- Modos de funcionamiento agrupados según operativas (en local, en remoto, en programación, etc.) para ello se recomienda agruparlos en forma tal que se correspondan con situaciones claramente diferenciadas (a través de unidad de visualización como ser pantallas, señales, ventanas, presentaciones luminosas, o de sonidos, acústicas, etc.)
- Para cada modo se definen las tareas, las cuales se identificarán de acuerdo a la nomenclatura definida y se seguirá siempre el mismo orden en las definiciones.

Las tareas para cada modo son:

- **Acción:** lista de acciones a realizar dentro del modo cuando se entra en un módulo, función, rutina, etc.
- **Evento:** Lista de eventos posibles con los elementos que definen su actuación (ejemplo, una interrupción temporizada). Para definir varios elementos se escribirá con una disposición sangrada. Debe considerarse que hay dos tipos de eventos: los normales y los temporizados. Para un evento normal, puede interesar: a) su invocación, b) su activación, c) permanencia en activación (medida en número de intervalos de tiempo (clock) que lleva activado y d) su desactivación, e) la permanencia desactivado, f) su destrucción y la correspondiente actualización de la información.
- Para un evento tipo temporizado interesa solo la activación con su correspondiente acción generada y la desactivación.
- Como observación podemos agregar que un evento puede ser una agrupación de subeventos (ejemplo, ingreso de datos por teclado o el pulsado de una secuencia de teclas específicas cuando se ingresa una contraseña (password)).
- **Periodo:** Lista de acciones periódicas fijas a realizar mientras se está en el modo.
- **Presentación:** acciones necesarias para actualizar la presentación al usuario en ese modo.

ASPECTO TÉCNICO

En cuanto al aspecto técnico, el software se hará siempre reubicable y estructurado. Estos dos requisitos facilitan la depuración, la comprensión y el mantenimiento de los programas.

PRESENTACIÓN

Recomendamos observar los siguientes puntos para la presentación de los programas:

1. Cada módulo, rutina, función, etc., deberá estar presentado de la siguiente forma:

- Encabezamiento: *(Entre símbolos de comentarios)*
  a) Nombre de la función, rutina, módulo, etc.
  b) Objetivo de la misma
  c) Parámetros de entrada
  d) Parámetros que devuelve
  e) Apellidos y nombres del o los autores
  f) Fecha de creación
  g) Fecha de última modificación o release
  h) Escueto comentario escrito sobre la modificación

- Comentarios documentados: En cada línea de codificación en que se realiza una operación compleja, se deberá incorporar un comentario a modo de documentar en pocas palabras lo que se pretende realizar con el código en ese renglón. Esto facilita enormemente la lectura y comprensión del código.

2. Los programas deberán estar estructurados en secciones o módulos bien separados en forma que sea totalmente modular y que facilite las tareas de depuración cuando se modifique una función o parte de ella y no requiere retocar todo el programa en su conjunto.

3. El lenguaje que se utiliza para la codificación tiene predefinido un conjunto de palabras llamadas palabras claves o palabras reservadas (que son las primitivas de dicho lenguaje) dado que solo pueden ser usadas en una determinada y particular forma.
Para construir los módulos se requiere cumplir cuatro pasos fundamentales, a saber:

- Primer paso: desarrollo de las especificaciones
  a) Estudiar el problema a resolver
  b) definir un nombre de la función, programa, etc.
  c) definir las interfases de entrada y salida que usará este módulo (variables globales)
  d) definir las variables locales

- segundo paso: desarrollo del programa
  a) programar el algoritmo que realizará el módulo

- tercer paso: prueba del programa
  a) funcionamiento
  b) depuración
  c) optimización

- cuarto paso: catalogación (no siempre se implementa)
  Consiste en incorporar el módulo en una estructura de forma tal que pueda ser fácilmente utilizable en otros códigos a través de un editor de enlaces (linkeditor).

4. En cuanto a los programas completos deberán considerarse
    a) Su instalación por única vez
    b) Su inicialización todas las veces que se lo invoca
    c) Su desinvocación

5. Y desde el punto de vista del usuario se deberá considerar todas las interfases que permitan un fácil uso e interacción con las presentaciones (teclas, mouse, pantallas sensibles, etc.)

ALGUNOS CRITERIOS DE CALIDAD PARA DISEÑAR SOFTWARE

Prevenir implica analizar los proceso para determina donde y por qué pueden ocurrir un mal funcionamiento de un programa o errores y luego generar ideas para planificar la mejor manera de evitar dichos errores. La prevención no debe ser solamente una herramienta práctica de trabajo sino una "actitud" y "estado de alerta" en todos los integrantes del grupo de trabajo. Para detectar las fuentes de errores y equivocaciones. Un buen ejemplo de búsqueda y hallazgo de errores lo encontramos en la filosofía oriental: **"Un Error es un pequeño Tesoro"** ya que esto nos permite mejorar. Sin ello no sería posible.

Esta preparación y disposición que se hace anticipadamente o durante el desarrollo del software permite resolver, como ya se aclaró anteriormente, desprolijidades y pérdidas de tiempos en mantenimientos y depuraciones posteriores, o modificaciones parciales y parches que pretenden hacer funcionar el programa durante la ejecución. Por todo ello, nuestra recomendación es: **PENSAR A PRIORI, LUEGO EJECUTAR LA TAREA.**

Cuando se diseña un software no debe aceptarse el criterio de que se lo realiza para "cumplir" con las exigencias o se hace "más o menos" o que se eviten los errores "casi siempre". Se debe ser original, resolver los algoritmos en forma simple, tener presente el objetivo de las especificaciones y pensar constantemente como mejorar la eficiencia tanto del trabajo como del resultado.

El avance y el perfeccionamiento en el desarrollo de un software es un ciclo continuo que se realiza en la siguiente secuencia:

1. El estudio previo del proceso o especificaciones
2. La generación de ideas
3. El diseño de un nuevo plan
4. La ejecución del plan
5. La observación y el análisis de los resultados
6. La generación de nuevas ideas a partir de la nueva situación
7. Ir al punto 3 mediante un análisis para mejorar el proceso.

Es de destacar que este proceso debe ser compartidos por todos los integrantes del grupo y en particular liderada hasta un límite en que cada integrante se sienta orgulloso de pertenecer al grupo y realice su tarea eficientemente y con satisfacción, entonces se puede afirmar que el trabajo individual está comprometida con la calidad del producto y este sentimiento de generosidad y confianza basado en la convicción de que el producto es patrimonio de todos y contribuye al logro de los objetivos comunes.

Por otro lado este proceso permite a cada integrante aumentar gradualmente el conocimiento del problema, efectuando un adecuado aprendizaje del mismo. No sirve como aprendizaje copiar otras soluciones, salvo que del análisis de estas soluciones surjan mejoras.

Para lograr una mejora continua (como hábito) en el proceso es necesario desarrollar la aptitud de investigación y análisis propia para aplicarle a cada problema nuevo. Sí es válido estudiar las distintas soluciones existentes para un dado problema. Con ello se inicia la secuencia de estudio y se puede extraer conclusiones que ahorran esfuerzos y tiempo.

El ciclo puede repetirse infinitamente, y pude inferirse que con ello se incrementaría la calidad del producto, pero no es así. El apropiado balance entre el costo y la calidad es la clave del éxito de la actividad desarrollada.

Todo el proceso de desarrollo de un software se pude subdividir en actividades elementales (módulos) definibles y cuantificables. Para determinar cuáles son las causas de ocurrencia de fallas y desviaciones es necesario actuar sobre estos procesos elementales.

Se ha planteado que la producción del software debe encararse modularmente en forma tal que cada interfase funcione con el concepto servidor /cliente como una cadena sucesiva hasta llegar al cliente externo final.

Cada cliente es una figura trascendente pues es el que recibe y fija los atributos como también las características en las especificaciones de cada módulo. Es aquí donde se debe implementar una adecuada estrategia para lograr los objetivos parciales y totales del producto.

La definición de los procesos elementales y el diagnóstico de la situación es la base fundamental en el desarrollo del software para lograr un adecuado nivel de calidad del mismo.

Se debe detectar o conocer cuáles son las actividades que no aumentan "valor agregado" de cada módulo o producto, entonces estamos en el límite del ciclo propuesto.

COMO RESUMEN AFIRMAMOS

- Para que el equipo funcione se requiere la participación de todos los individuos mancomunados y con objetivos comunes. La motivación y la actitud de los integrantes del equipo es un factor clave para lograr este objetivo de trabajo participativo. Como observación agregamos que la actitud humana no es algo inmodificable, sino que está estrechamente ligada a las características y el clima (ambiente) en que se vive para desarrollar la tarea.
- Es necesario movilizar toda la inteligencia y aprovechar todas las ideas de cada integrante
- No se puede garantizar el éxito de los objetivos si no se participa con responsabilidad
- Debe existir una actitud individual predispuesta a la cooperación y el predominio de los intereses generales por sobre los intereses particulares o individuales. Sin ello no se logra el trabajo en equipo y tampoco se puede coordinar esfuerzos.
- Para el desarrollo del trabajo en equipo es necesario que se den las siguientes condiciones:
    1. Actitud de comprender los problemas
    2. Actitud de cooperación y confianza entre las personas del equipo
    3. Una formación general sobre la problemática encarada (comprensión del problema)
    4. Una formación especializada (manejo de lenguajes, códigos, etc.)
    5. Una actitud permanente de pensar, planificar, investigar y analizar las situaciones

Esta pequeña introducción al diseño de software no pretende ser completa y solo presenta una visión personal de este complejo campo ingenieril.

NORMAS PARA LA PRESENTACIÓN ESCRITA DE LOS T.P.

Introducción

Este documento tiene por objetivo dar algunas guías y sugerencias que ayudarán a los alumnos a producir sus trabajos prácticos con un alto grado de cuidado, precisión y elegancia. También pretendemos homogeneizar la presentación de los trabajos solicitados durante el curso. Los siguientes lineamientos son aplicables a todos los trabajos prácticos propuestos.

Las normativas que a continuación se enumeran tienen carácter de **complementario**, es decir que, bajo ningún punto de vista se deberá usar el presente documento como refutación o reemplazo de las normas vigentes establecidas por la Universidad. De la misma forma, las normas establecidas en el documento de planificación de la Cátedra, tienen carácter de marco regulatorio en vista a las normativas para este curso.

La presente documentación está sujeta a cambio, es responsabilidad del alumno notificarse de las alteraciones producidas. Para tal fin, las versiones que surgieren durante el transcurso del año lectivo estarán disponibles en las siguientes locaciones:

Directorio de la Cátedra, Server del Laboratorio de la Universidad.

Lineamientos Generales

Para la correcta presentación de cualquier trabajo se deberán seguir las siguientes pautas generales:

a) Carátula o Tapa de Encuadernación

Al comienzo del trabajo deberá figurar una carátula (normalizada según documento provisto por la cátedra) debidamente completada.

b) Escrito, formato de contenidos

Para la presentación del trabajo deberán utilizarse dos medios de almacenamiento distintos. El primero es en papel tamaño A4, o carta (no oficio). El segundo en formato digital, referirse al apartado "***Reglamento de entrega y reentrega de Trabajos prácticos***" para más información. Todos los escritos y representaciones gráficas deberán ser impresos y el disco deberá contener los archivos de datos que generan las impresiones y los programas correspondientes. De ninguna manera serán aceptados los trabajos manuscritos, salvo en las revisiones **informales**.

***NOTA**: el equipo de trabajo debe garantizar que el material entregado, en cualquier tipo de medio magnético u óptico, está **libre de virus y defectos**. La no observancia de esta normativa implicara un grave daño para la evaluación del material entregado.*

La tipografía a utilizar deberá ser clara, de fácil lectura y compatible con sistemas OCR (Optical Carácter Recognition), sugiriéndose: letra de imprenta, en cualquiera de sus estilos (preferentemente Arial o Roman), no aceptándose letras cursivas, ni góticas. El tamaño de letra a utilizar para el cuerpo de texto, deberá ser de 10 a 12 puntos. El cuerpo de texto no podrá ser todo en mayúsculas. Todos los comienzos de párrafo deberán tener una sangría de ocho espacios al margen izquierdo y justificados a ambos márgenes del documento.

Deberá incluirse un pie y encabezado de páginas en todas las hojas, menos en carátulas. El pie de página deberá contener el número de página como mínimo, en cursiva y con el prefijo “Página”. El encabezado deberá contener el título del trabajo y el nombre del equipo, en un tamaño de letra que no difiera con el del texto y en negritas.

c) Centrado del Documento

El material o contenido debe estar dispuesto simétricamente con referencia al centro del texto escrito y no equidistante de los bordes, el margen izquierdo deberá ser mayor al derecho para permitir la encuadernación. Los dibujos, tablas, gráficos y demás objetos incrustados, también se colocarán con referencia al centro del texto y no de la hoja en sí. El centrado vertical deberá ser el conveniente para la correcta impresión y visualización de los encabezados y pies de páginas.

d) Márgenes y sangrías

De acuerdo a las Normas IRAM para el formato A-4 corresponde unos 20 mm como mínimo para el margen izquierdo, y 20 mm para el derecho, 25mm para el borde inferior de la hoja y desde el octavo renglón del borde superior. Es aconsejable respetar esta normativa, sin embargo es posible usar estos parámetros en función de la correcta visualización del material. Las sangrías deberán ser respetadas a lo largo de todo el documento, utilizando la capacidad de tabulación automática de algunos procesadores de texto.

e) Separación de Palabras, micro justificación

Si es necesario realizar la división de palabras, debe hacerse siempre entre dos sílabas. Los números, fórmulas, fechas, y nombres propios no se dividirán. La micro justificación de los caracteres deberá ser fijada en forma automática.

f) Alineación de Enumeración de Títulos y Subtítulos

Los encabezamientos deben cumplir un doble fin: poner título a la sección o grupo de párrafos y facilitar la consulta rápida. Para ello se establece la siguiente estructura de los encabezamientos:

1. Titulo principal o Nivel 1
  1.1. Subtítulo o Sección Nivel 2
    1.1.1. Enumeración Temática o Nivel 3
    1.1.2.
    1.1.3.
  1.2.
    1.2.1.
      1.2.1.1. Categorización de contenidos o Nivel 4
      1.2.1.2.
    1.2.2.
  1.3.
2.

Cuando deben alinearse números arábicos o romanos deberá procederse de la siguiente forma: (alineando la cifra final y no la primera)

9	(i)	I
12	(ii)	II
111	(iii)	III

Las carátulas, el índice y la Introducción deberán numerarse en formato romano, exceptuando de ello a la carátula principal. De la misma forma se deben numerar los capítulos y secciones del documento. Para tal fin debe crearse los estilos correspondiente en la galería de estilos del procesador de textos, e incluirlos como marcadores para la generación del índice de contenidos.
Las carátulas no llevarán encabezamiento ni pie de página.

g) Espaciado Vertical

Todo el texto del documento debe ser escrito a un espacio. El título inicial se coloca a cinco espacios del margen superior o del encabezamiento y luego a tres espacios se comienza a escribir. Los encabezamientos importantes se colocarán a tres espacios antes y a tres después; los demás títulos tienen tres espacios antes y dos después.
Los capítulos deberán indicar en su encabezamiento, en las páginas pares el título del trabajo, y en las impares, el capítulo y el tema del cual trata. En cada pie de página figurará el número de página correspondiente, numeradas en forma consecutiva con números arábigos. Tanto el encabezamiento como el pie de página, deberán estar separados del área de texto por una línea trazada desde el margen izquierdo al margen derecho.

h) Títulos, Subtítulos y Secciones Destacadas

Los títulos principales o de Nivel 0, tendrán que figurar centralizados, subrayados, con letra itálica y resaltada, con un tamaño que oscile entre los 20 y 30 puntos. Los subtítulos deberán ir numerados, con el número de capítulo, un punto y el número de sección, se ubicarán desde el margen izquierdo con sangría explicada en el punto f), subrayados y en letra resaltada. Para éstos se utilizara un tamaño de letra entre los 12 y 16 puntos.
Los títulos de párrafos figurarán sangrados desde el margen izquierdo, con el mismo tamaño de letra que se utiliza en el cuerpo principal, subrayados y en negrita. Deberán estar numerados con el número de capítulo, el número de sección y el número de párrafo, separados por puntos.

Para resaltar palabras o frases dentro del cuerpo de texto se podrán utilizar tanto el resaltado como la letra itálica, siempre que se respeten el tamaño y el tipo de letra utilizado en el cuerpo de texto.
Cuando se deba enumerar un contenido, éste tendrá que estar tabulado y para la enumeración se utilizarán números arábigos, separados por un guión del texto.
Los cuadros o gráficos (en lo posible), serán escritos en el texto directamente.

i) Del Índice General

El título del mismo deberá figurar como cualquier título del resto del trabajo. Cada capítulo figurará al margen izquierdo, con su correspondiente número, separado por un guión del nombre, subrayado y en letra resaltada, y no deberán llevar número de página.
Los subtítulos figurarán en un primer grado de tabulación, con el número formado de la misma manera en que se colocaron en el resto de la obra, y a continuación el nombre en letra resaltada. En caso que un tema no quepa en un solo renglón, su continuación, deberá estar alineada con la primer letra del nombre del tema, además, el número de página (si correspondiese), en que se encuentra, deberá figurar en el último renglón y alineado al margen derecho.
Los títulos de párrafos figurarán en un segundo grado de tabulación, con el mismo número que tienen en el cuerpo del texto, con letra resaltada y cumpliendo los términos que se detallan en el párrafo anterior.
Los apéndices figurarán como capítulos independientes, pero no llevarán numeración en sus subtítulos y títulos de párrafos. De la misma forma, figurará la bibliografía.

j) Los Apéndices o Anexos

Los apéndices contendrán información relacionada con el tema principal del trabajo. En ellos figurarán todos aquellos datos y documentos que complementen el texto, y que figurando en el cuerpo principal harían dificultosa su lectura.
Los apéndices deberán enumerarse alfabéticamente, los títulos, subtítulos y títulos de párrafos cumplirán las mismas normas que las del cuerpo principal con la excepción de la numeración.
Cada uno de los apéndices en lo posible, deberá presentarse con un breve encabezamiento, o si el alumno lo prefiere, una carátula en la que figure, el número de apéndice y el tema tratado.
La numeración de las páginas de un apéndice será correlativa a la del resto del trabajo.

k) La Bibliografía

En ella deberá figurar todo aquel material que se haya consultado o estudiado para la confección del documento del trabajo práctico. Se detallará en letra resaltada el título de la obra, seguido por el / los autores en letra normal y en los renglones siguientes, en forma tabulada, la editorial, año de edición y demás datos complementarios. La bibliografía siempre se coloca al final del documento.
También se deben incluir los URL de los sitios de Internet consultados.

l) Encuadernación

Los trabajos serán presentados adecuadamente encuadernados o encarpetados. Este puede ser con carpetas tipo plásticas, con carpeta transparente y lomo de plástico, con anillado o tipo libro con tapas ad hoc. El método de encuadernación utilizado deberá contener los aditamentos necesarios para la contención del medio magnético u óptico asociado al trabajo.

m) Identificación:

Todas las hojas presentadas, en trabajos de teoría o prácticas, (salvo la carátula) deberán contener en el Encabezado o Pie de página el nombre del autor o grupo, curso, fecha de presentación y nombre del trabajo. El tamaño de letra a utilizar para el encabezado, deberá ser de 8 a 10 puntos.
NO SE ACEPTARÁN HOJAS SUELTAS.
Los Archivos en los soportes ópticos – magnéticos se identificarán de la siguiente forma:
<Apellido del Alumno (o grupo)_Siglas de la Universidad_Curso_año_Título de la entrega>
Ejemplo: “PEREZ UNLAM Com35 2014 monografía sobre Device Drivers”
Todas las presentaciones de trabajos teóricos en medios ópticos o magnéticos deberán tener una etiqueta rotulada con el Nombre del Trabajo o Número de presentación, Apellido y Nombre del Alumno (o todos los integrantes del grupo), Universidad Curso, año, título.
La presentación de Monografías, deberán estar contenidas en un solo archivo.

Consideraciones especiales

A continuación se detallan el uso de algunas características especiales para la presentación de los trabajos prácticos. Solo deberán ser tomadas en cuanta en las situaciones convenientes.

a) Uso de campos especiales

Algunos procesadores de texto permiten la inclusión de campos especiales, como ser campos de datos o formularios. Siempre que sea conveniente, la inclusión de estos campos deberá estar documentada en un apartado. La experiencia indica que “sobrecargar” un documento con estas características puede ser contraproducente, sin embargo a los efectos de comprensión del tema expuesto es posible que se presente la necesidad de uso de estas facilidades.
Se deberá utilizar un utilitario editor standard en el mercado para la inclusión de campos especiales como ser, archivos de sonidos, gráficos, u otros datos que puedan adjuntarse al documento. Es aconsejable el uso de herramientas pertenecientes a un mismo paquete.

b) Galería de estilos

Cada documento deberá tener una galería de estilos utilizados. El objetivo de esta facilidad es mantener en forma uniforme el formato del documento. Es aconsejable que cada equipo genere un standard a utilizar en todos los documentos que entregue. Cada uno de los equipos deberá generar un archivo de descripción de estilos (.dot en el caso de Microsoft Word) y adjuntarlo a cada uno de los documentos.

c) Software sugerido

La siguiente lista de software tiene carácter de sugerencia debido a que son estándares del mercado en la actualidad. Cualquier otro aplicativo que el equipo de trabajo desee utilizar deberá ser provisto a la cátedra en su debido tiempo y forma.

- Procesador de Texto, Microsoft Word 97 o superior
- Aplicativo de Presentaciones, Microsoft PowerPoint 97 o superior
- Planilla de cálculos, Microsoft Excel 97 o superior
- Bases de Datos, cualquier motor que respete la norma DBASE III
- Generador de Páginas HTML, Microsoft FrontPage
- Aplicación para graficación de Esquemas, Visio Tech 4.0 o superior

Es aconsejable el uso de herramientas desarrolladas con fines específicos, como es el procesador de texto LATEX.

d) Formato del medio óptico o magnético

Se deberá generar la siguiente estructura para la entrega del material. La etiqueta del medio deberá contener el nombre del grupo y el título del trabajo entregado, la etiqueta que se genera con el formateo del medio debe contener el número del grupo y la clave de la entrega (nn-clave), según las siguientes normas:

Nn: número del equipo de trabajo

Clave: <F|R><ff> donde F: Final, P: Revisión, ff:fecha empaquetada

La estructura de directorios deberá ser la siguiente:

\<Raiz>
Documentación
Manual de Uso
Desarrollo Temático
Documentación de Soporte
Fuentes
Ejecutables
Aplicativo
Versión 1
Versión 2
Versión n
Utilidades
Editores
Bibliografía

Reglamento particular de Entregas

Esta guía de Trabajos Prácticos consta de cuatro trabajos todos de carácter grupal, que son obligatorios para la aprobación de la parte práctica de la materia, y que junto a la aprobación de la parte teórica (parciales / parcialitos), formarán la nota final de la materia Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)

Introducción

Objetivos

1. Generar un marco de trabajo para la puesta en marcha de diversos aspectos teóricos de las materia Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)
2. Desarrollar actividades que faciliten la adquisición de nuevos conocimientos en el ambiente de Sistemas Operativos

Alcances

Todos los módulos de la materia presentados en la guía introductoria.

Estas Normativas Complementarias son aplicables a todas las normas y procedimientos establecidos en la Universidad para el desenvolvimiento de un curso práctico.

Son aplicables todas las normas y procedimientos internos de la Cátedra Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)

Rol del Equipo de Trabajo

Para los trabajos prácticos grupales, los alumnos deberán formar grupos de trabajo con un mínimo de una persona (no es recomendable, pero admitido), y un máximo de 5 (cinco) alumnos. Todos los integrantes de un grupo tienen que pertenecer a la misma comisión, no estando permitido formar grupos con integrantes de más de una comisión.

A diferencia de años anteriores, los grupos formatos tendrán validez solamente por un trabajo práctico, lo que permite a los alumnos a cambiar de grupo entre un trabajo y otro. Para tal fin, el alumno deberá presentar al inicio del trabajo práctico un formulario indicando cual será el grupo con el que trabajará para dicho trabajo práctico.

Cabe aclarar que en ningún caso se permitirá cambiar de grupo en el medio de un TP., y que en caso de tener que reentregar / recuperar un trabajo práctico luego de un cambio de grupo, lo deberá realizar con el grupo original. Esto quiere decir que el grupo conformado para un determinado trabajo práctico es inmutable, y deberá ser concluido por los alumnos que continúen la cursada dentro de las fechas reglamentadas.

Rol de Líder de Equipo

Una vez formado el equipo de trabajo, deberán elegir un líder de grupo. Los mecanismos para la selección serán de entera responsabilidad del equipo en cuestión. Una vez fijado el miembro líder, este mantendrá su rol hasta finalizado el práctico. Este miembro es el responsable de generar el "Reglamento de Convivencia" que debe ser observado por el resto del grupo. El desempeño de este rol no tiene ningún tipo de impacto en vista a las evaluaciones grupales. Las responsabilidades "extras" de este rol serán fijadas oportunamente en cada curso.

Rol de Supervisor

Este rol será llevado adelante por un docente responsable de la práctica de la materia, asignado al grupo en cuestión. El objetivo de este rol es suministrar un lineamiento general en el desarrollo de los temas asignados a un equipo determinado.

Rol de Revisor

La cátedra nombrara un docente en carácter de revisor para la evaluación individual de cada trabajo entregado.

Mesa de Examen

El Jefe de la cátedra o el revisor presidirá la evaluación final con todo el equipo docente que haya intervenido en el desarrollo de los trabajos.

Mecánica de los Trabajos Prácticos.

Las normas redactadas a continuación tienen por objetivo fijar un marco de trabajo que permita un desarrollo ordenado de las prácticas.

Modalidad de Trabajo

La modalidad de trabajo para la práctica será la de "trabajo en equipo supervisado", modalidad que imprime un ritmo dinámico a las clases prácticas. La responsabilidad de desarrollo de la práctica recae en el equipo en general y por igual a todos sus miembros, sin considerar roles especiales de los mismos (líder de grupo). El equipo de trabajo retiene en su esfera de actividades todas las cuestiones relacionadas a la administración del tiempo y el planeamiento para la calidad. Toda problemática, que no sea estrictamente relacionada con la temática a desarrollar, será discutida y resuelta en forma interna en el equipo de trabajo.

Evaluación del Trabajo Práctico.

Ámbito:

Es aplicable a todos los cursos de la cátedra de Sistemas de Computación II (Plan 1997) / Sistemas Operativos (Plan 2009)

Evaluación de Contenidos

La evaluación de contenidos se realizara sobre la base de lo expuesto en la guía de la materia.

Evaluación de Cohesión

La evaluación de cohesión tiene como objetivo ponderar el trabajo de cada uno de los individuos como miembros de un equipo.

Evaluación de Exposición y Defensa

La evaluación de exposición y defensa tiene como objetivo ponderar al equipo y a sus miembros en la claridad y solidez con que exponen los temas desarrollados. Así mismo, se evaluará la solidez de las demostraciones para respaldar las teorías expuestas.

Evaluación de Capacidad de Crítica

La evaluación de capacidad de crítica tiene como objetivo ponderar al equipo y a sus miembros en la capacidad de crítica frente a la exposición de otro equipo o dentro del suyo propio. Solo serán permitidas las críticas al modelo presentado, bajo ningún punto de vista será permitida una crítica a la mecánica o constitución de otro equipo de trabajo. Esta evaluación se aplicará en los cursos o grupos que decidan exponer sus trabajos a la crítica de otros grupos.

Evaluación y Calificación Finales

El conjunto de evaluaciones parciales, arriba expuesto, será fuente de información valiosa para la estructuración de una calificación y referencias finales.

Actividades Prácticas

Entregas de Trabajos

Durante el ciclo lectivo, las actividades prácticas propuestas resultarán en un "Entregable"

Se observarán dos tipos de entrega:

Entregas de Revisión

Pueden ser efectuadas en cualquier momento, sin importar el orden de secuencia de los temas a desarrollar. Deberán observarse todas las normas que estén vigentes para las entregas. La frecuencia de estas entregas es la estipulada en cada TP., no pudiéndose realizar más de una por semana. Tiene carácter obligatorio la primera entrega de revisión al promediar la mitad del periodo previsto para el tema en cuestión. El resto de las entregas es de carácter opcional. Las evaluaciones de los trabajos entregados, en carácter de revisión, son tomadas en cuenta para la evaluación del trabajo final.

Entregas Finales

Son de carácter obligatorio, en tiempo y forma. La no entrega de un trabajo final genera su desaprobación automática. Se deberán observar las fechas y horarios de entrega, los trabajos que superen ese lapso de tiempo serán recibidos en calidad de recuperatorios.

En caso de no entregarse en término, como penalidad, se asignarán nuevos trabajos prácticos de mayor dificultad. Todos los Trabajos Prácticos deberán estar aprobados en la fecha prevista para la firma, en caso de no hacerlo, se asignará un trabajo extra de Recuperatorio que el alumno deberá completar y entregar funcionando en la fecha que se determine para cada caso. Esta entrega se efectuará durante los días fijados para ello y constará, además de los Trabajos Prácticos adeudados, de un examen el recuperatorio integrador de la materia. Cabe aclarar que la dificultad del trabajo práctico adicional será mucho mayor que la de todos los Trabajos Prácticos asignados durante la cursada.

Los Trabajos Prácticos, deberán ser entregados en los medios indicados en cada trabajo, respetando para todos ellos el reglamento de presentación de TP, como ser un mismo tipo de letra, formato de papel, etc.

En las carátulas de presentación de todos los Trabajos Prácticos, se deberá consignar, número de trabajo práctico y versión de entrega y los nombres, apellidos y

matrículas o legajos de todos los integrantes del grupo. En caso de entregar medios de almacenamiento, en la etiqueta de los mismos, deberán figurar los mismos datos que en la carátula. Si se tratase de una entrega complementaria, se deberá anexar una descripción de las modificaciones efectuadas con respecto a la entrega anterior. No se aceptarán entregas que no cumplan las condiciones solicitadas, esto quiere decir, que si la presentación se debe realizar en disquete, y papel, no se aceptará que se entregue uno, y no el otro (Ej. el disquete solo, o la carpeta sola).

En los trabajos entregados en medios magnéticos se deberá tener especial cuidado en la presencia de virus informáticos y posibles errores en el medio de almacenamiento, en caso de existir no se considerará entregado dicho trabajo y el mismo deberá ser RECUPERADO. También se debe tener en cuenta, los posibles errores de lectura de las unidades magnéticas, para esto se recomienda grabar el medio en una unidad, y probar copiarlo desde otra.

Los materiales entregados pasan a ser propiedad de la cátedra, por lo cual se pide tener especial cuidado en no dejar en los medios a entregar información de otras materias, o importantes para el alumnado, y que no tiene que ver con la materia, y/o trabajo práctico entregado.

También se debe tener en cuenta que los docentes de la cátedra, se toman el trabajo de comparar el código fuente de cada uno de los trabajos con los de otros grupos, comisiones e incluso años, por lo tanto se les recomienda no copiarse los trabajos, puesto que en caso de existir fraude, se perderá automáticamente el trabajo, y se les incorporarán a los grupos intervinientes en dicho fraude, sendos trabajos prácticos, los que deberán ser entregados junto con el trabajo falseado, efectuado nuevamente. Cuando se indica “grupos intervinientes”, se considera tanto el grupo que copió el trabajo como el grupo al que le pertenece el trabajo copiado. Por esto se sugiere tener especial atención con los trabajos que se dejan dentro de las cuentas de los servidores de la Universidad, y sobre todo con las passwords seleccionadas, el cambio periódico de las mismas y su confidencialidad.

Defensa del Trabajo Entregado

La defensa de un trabajo tiene carácter obligatorio y consiste en responder todas las cuestiones formuladas por los docentes que evalúan los trabajos entregados. La defensa del trabajo se realizará en el horario y fecha preestablecido por los docentes responsables de estas actividades.

Exposición y Defensa de Trabajos Prácticos

Ámbito:

Esta modalidad es válida para aquellos cursos que decidan que cada grupo exponga su trabajo a la crítica de los restantes grupos.

Criticas a Trabajos

La crítica a los trabajos de otros equipos se regirá por la dinámica grupal en una clase práctica. Todas las críticas deberán estar debidamente fundamentadas y posteriormente escritas, pues deben figurar en el informe final de la presentación.

Defensa del Trabajo

La defensa de un trabajo consiste en escuchar y aceptar las críticas de otros grupos de pares que enriquecerán la posterior refutación y/o reconocimiento de las propuestas de cambio. En la dinámica grupal las refutaciones pueden ser orales, siendo necesario transcribirlas posteriormente en el informe final de la presentación. La defensa del trabajo se realizará en el horario y fecha preestablecido por los docentes responsables de estas actividades.

Publicaciones

Trabajos a Publicar

Todos los trabajos realizados están sujetos a publicación en la biblioteca de la Universidad. El objetivo es poner a disposición de los demás alumnos los trabajos realizados, como fuente de información y referencia futura.

Derechos

Se deja constancia que la Cátedra se compromete a no hacer usufructo comercial de los desarrollos realizados por los alumnos. Sin embargo, se reserva el derecho de utilizar dichos desarrollos para futuros prácticos que permitan enriquecer los resultados que se obtengan. Al finalizar el presente cuatrimestre se podrán seleccionar los mejores trabajos para tomar como base de prácticos de futuros cursos como también su posible utilización (sin lucro) por parte de la Universidad, con la debida mención de sus autores.

Condiciones para la Publicación

Si bien todos los trabajos pueden ser publicados, al finalizar el año se seleccionarán los trabajos de mejor calidad. Los factores a tomar en cuenta son los mismos para las evaluaciones. Otra condición de publicación del material desarrollado es la aprobación por parte del director de la cátedra y de una mesa examinadora especialmente formada para tal fin.

Mecanismos de Publicación

Los trabajos seleccionados deberán ser adecuadamente encuadernados por el equipo de trabajo y se suministraran dos copias idénticas de los mismos. Las dos copias serán donadas a la biblioteca de la Universidad y se permitirá el préstamo en calidad de material de consulta. Todas las normas para entregas se aplican a las copias de publicación.

Calendario de Entregables

A determinar oportunamente por los docentes responsables de las cátedras. En general se informara el calendario en cada curso.

Descripción de los Trabajos Prácticos

Trabajo Práctico Nro. 1:

Programación de scripts básicos, con awk y sed sistema operativo GNU/Linux (Grupal)

Trabajo Práctico Nro. 2:

Programación en ANSI-C, o C++, bajo sistema operativo GNU/Linux, para resolver ejercicios de concurrencia, manejo de hijos, señales y FIFOS (Grupal)

Trabajo Práctico Nro. 3:

Programación en ANSI-C, o C++, bajo sistema operativo GNU/Linux, para resolver ejercicios de manejo de semáforos, memoria compartida, Threads y Sockets (Grupal)

Trabajo Práctico Nro. 4:

Programación en ANSI-C, o C++, bajo sistema operativo GNU/Linux, de un ejercicio integrador basado en el desarrollo de un juego para consola gráfica (Grupal)

Defensa de TP's (Coloquio) y recuperatorio de parciales:

El recuperatorio de parciales será coordinado por los docentes con los alumnos durante el transcurso de la cursada.

Los recuperatorios de los trabajos prácticos y coloquios serán a partir del 24 de Noviembre de 2014, en el horario de cursada

Notas importantes:

- Todos los trabajos prácticos pueden ser incluidos como temas de parciales (desde el día de la fecha de entrega y sin importar si el mismo ya fue o no evaluado por los docentes de la cátedra).
- En caso de que los trabajos no se encuentren entregados en la fecha que se indica en el trabajo práctico (sin tener en cuenta la fecha de entrega tardía), no se podrán rendir los parciales o parcialitos hasta que el alumno entregue todos los trabajos adeudados.
- Si un trabajo no es entregado en la fecha correspondiente, se contará con un período de 7 (siete) ó 15 (quince) días corridos para su entrega (denominada fecha tardía de entrega – que se encuentra aclarada en cada trabajo práctico-). Una vez transcurrido el plazo no se aceptará la entrega hasta la fecha de recuperación.
- Para los trabajos prácticos que tengan fechas de entregas graduales, se considerará cada entrega parcial como un trabajo práctico por separado. De forma que si un trabajo práctico está dividido en dos entregas, serán como dos trabajos independientes pero que conformarán una sola nota; y para la nota final del trabajo práctico se tomará el promedio de las notas de las dos entregas.
- No se podrá entregar un trabajo práctico sin que se encuentre entregado el anterior, y aprobados todos los trabajos prácticos previos al anterior. Ej.: Si se quiere entregar el trabajo práctico número 3, se deberá tener presentado para su evaluación el trabajo práctico 2, y aprobado el trabajo práctico 1.

Reglamento de entrega y reentrega de Trabajos prácticos

Todos los trabajos prácticos tienen el mismo formato de entrega o reentregas que es descrito a continuación, y que en caso de no ser cumplido no será considerado como entregado el trabajo práctico.

Entrega escrita:

Por cada entrega o reentrega de trabajo práctico se deberá entregar una carpeta o folio conteniendo:

- **Una (1) ó Dos (2) carátulas** (dependiendo de lo que indique el docente del curso) utilizando para tal fin la plantilla disponible en el sitio web de la cátedra debidamente completada.
- Para poder realizar la entrega de cualquier trabajo práctico se deberá cumplir con las siguientes condiciones para cada uno de ellos:

TP	Entrega en fecha	Entrega tardía	TPs entregados	TPs Aprobados
1	Ejercicios 1 a 5	Ejercicios 1 a 6	**--**	**--**
2	Ejercicios 1 a 5	Ejercicios 1 a 6	**TP 1**	**--**
3	Ejercicios 1 a 4	Ejercicios 1 a 5	**TP 1 y TP 2**	**TP 1**
4	Juego funcionando	Juego funcionando	**TP 1, TP 2 y TP 3**	**TP 1 y TP 2**

Entrega digital:

En el directorio Home de cada usuario existirá una estructura de directorios que se utilizará como lugar de entrega de los ejercicios resueltos en cada trabajo práctico. Tenga en cuenta que cada grupo sólo necesita copiar el contenido en el Home de **uno** de sus integrantes.
La estructura de directorios será como la siguiente:

Entregas/
TP1/
TP2/
TP3/
TP4/

Esta estructura sólo está pensada para las entregas, no debiéndose utilizar para guardar archivos temporales o versionado de ejercicios que no estarán presente en la entrega final del trabajo práctico.

Cuando un determinado trabajo práctico se encuentre en condiciones de ser entregado, los alumnos deberán copiar todos los contenidos a entregar al directorio correspondiente y se deberá generar un archivo con el nombre ENTREGABLE (todo en mayúsculas) en el directorio que corresponde al TP. Este archivo servirá para informar al cuerpo docente que el TP está listo para ser evaluado.
Es importante destacar que este archivo será el que determine la fecha de entrega de su trabajo.

Un sistema preparado por los docentes generará todas las noches una copia de los TP's disponibles para ser entregados a un directorio del cuerpo de docentes, y renombrará el archivo ENTREGABLE colocándole la fecha de entrega como constancia de que el TP fue tomado para su corrección correctamente o un mensaje de error en caso de que su entrega no pueda ser realizada por algún problema técnico o de mala administración del directorio de entregas. Dentro del archivo se informará el error detectado.

Este mecanismo será utilizado también para las reentregas de trabajos prácticos.

Guía orientativa para el uso de GNU/Linux

- ***Descripción:*** A continuación se detallan una serie de preguntas y ejercicios orientados al uso de una terminal de caracteres de un ambiente multiusuario y multitarea basado en GNU/Linux. La intención buscada con esta guía es que aquellos alumnos que no estén familiarizados con una terminal de caracteres ni con la familia de sistemas operativos GNU/Linux, puedan aprender las principales características y poder realizar los trabajos posteriores con un mejor conocimiento del entorno en el que se deben realizar.
- **Nota:** Si bien este no es un trabajo práctico de entrega obligatoria, aquel alumno/a que crea conveniente realizarlo y desee consultar a los docentes o ayudantes de la materia sobre su contenido podrá hacerlo en cualquiera de las clases prácticas. Desde el cuerpo docente recomendamos a todos aquellos alumnos que nunca hayan trabajado en un ambiente de este tipo realicen esta guía y consulten a los docentes sobre los temas aprendidos y los no comprendidos
- ***Formato de entrega:*** sin entrega o con entrega de consulta optativa
- ***Preguntas:*** A continuación se detallan todas las preguntas y ejercicios que deberán ser resueltos. Tenga en cuenta que salvo en los momentos que indica que debe estar sesionado como **root**, en el resto de los ejercicios debe estar conectado como usuario común. (TIP: se le recomienda que primero realice todo el trabajo, anotando los resultados en papel, y a mano, y luego lo pase con el editor vi).

1. INTRODUCCIÓN

1.1. ¿Qué es la cuenta de superusuario (root) y para qué se utiliza?

1.2. Ingresar al sistema como superusuario (root), y realizar los siguientes pasos (éste punto no puede ser realizado en el laboratorio 266):

- 1.2.1. adduser <apellido> (reemplazar <apellido> por el suyo).
- 1.2.2. passwd <apellido> (Ingrese una contraseña (password) a su elección).
- 1.2.3. logout

1.3. Indique claramente qué efectuaron estos comandos, e indique qué archivo/s fueron modificados (Dentro del directorio /etc) **TIP**: Utilice lo siguiente: "ls -lt /etc | more".

1.4. Luego ejecute "cat /etc/passwd | more" y haga lo mismo con los otros archivos que se modificaron. Analice y comente lo visto

1.5. ¿En qué directorio se encuentran los comandos utilizados en los puntos 1.2.1, 1.2.2, 1.2.3, 1.3, y 1.4?

2. AYUDA

2.1. INFO: Info es un programa para leer documentación. Este se compone de una estructura del tipo árbol, dividido en nodos de información. Cada nodo describe un específico tópico con un determinado nivel de detalle.

2.1.1. Ingrese a info y responda:

- 2.1.1.1. ¿Cómo se llama el nodo raíz de Info?.
- 2.1.1.2. Ubique el cursor en la línea (* cp:) y presione ENTER.
- 2.1.1.3. ¿Qué sucedió?.
- 2.1.1.4. ¿Cómo se llama este nodo?.
- 2.1.1.5. ¿Cuál es el próximo nodo?.
- 2.1.1.6. ¿Cómo puedo moverme al próximo nodo?.
- 2.1.1.7. ¿Cómo puedo moverme al nodo anterior?.

2.1.2. Presione la tecla 'u'.

- 2.1.2.1. ¿Qué sucedió?.
- 2.1.2.2. ¿En qué nodo se encuentra?.

2.1.3. Repita el punto 2.1.2. hasta que llegue a la raíz de Info.

- 2.1.3.1. ¿Con qué tecla puedo volver directamente a este nodo?.
- 2.1.3.2. ¿Cuál es el método directo para acceder al nodo cp?.(tip: sin desplazar el cursor).

2.1.4. ¿Cómo puedo buscar una palabra clave dentro de un nodo?.

2.1.5. ¿Cómo puedo buscar la siguiente palabra clave, buscada anteriormente?.

2.1.6. ¿Cómo puedo salir de Info? - salga.

2.2. MAN: man es un programa que formatea y muestra la páginas del manual.

2.2.1. ¿Cuál es la diferencia entre man e info?

2.2.2. ¿Cómo puedo ver la información de un determinado comando?

2.2.3. ¿Cómo puedo buscar una palabra clave dentro de la página del manual?

2.2.4. ¿Cómo puedo salir?

2.2.5. ¿Cómo hago para buscar una palabra clave determinada en todas las páginas del manual?

2.2.6. ¿Qué es lo sucede al realizar lo siguiente?

- 2.2.6.1. man
- 2.2.6.2. man man
- 2.2.6.3. man cp

2.2.6.4. man printf
2.2.6.5. man fprintf
2.2.6.6. man sprintf
2.2.6.7. man cd
2.2.6.8. man 3 printf

2.2.7. Del punto anterior, responder:

2.2.7.1. Al invocar man junto con fprintf y sprintf muestra la misma página. ¿Por qué no muestra la misma página al invocarlo con printf?. (TIP: vea el punto 3.2.6.2).

2.2.7.2. ¿Cómo puedo invocar al man para ver directamente la función printf del lenguaje C?.

2.3. HELP: help es la ayuda que ofrece el shell de GNU/LINUX para utilizar sus comandos.

2.3.1. ¿Cuál es la diferencia entre help e info?.
2.3.2. ¿Cuál es la diferencia entre help y man?.
2.3.3. ¿Qué sucede al invocar al help?.
2.3.4. ¿Cómo puedo ver la información de un determinado comando?.

2.4. whereis

2.4.1. ¿Qué sucede al utilizar el comando whereis cd?
2.4.2. ¿Qué es la información que se muestra por pantalla al ejecutar el punto anterior?
2.4.3. ¿Qué ocurre si se ejecuta whereis * sobre un directorio? (**Tip**: si no pasa nada, intentelo nuevamente pero primero ejecute cd /bin)
2.4.4. ¿Cuál es la diferencia entre whereis y find?

2.5. whatis

2.5.1. ¿Qué sucede al utilizar el comando whatis cd?
2.5.2. Si el resultado del punto anterior fue la leyenda “cd: nothing appopiate”, utilice el comando /usr/sbin/makewhatis, y responda los siguientes puntos:
2.5.2.1. ¿Qué realizó la sentencia anterior?
2.5.2.2. Reintente el punto anterior.
2.5.3. Cambie al directorio /bin, y ejecute el comando whatis * ¿Qué ocurrió?
2.5.4. Utilice el comando apropos passwd y whatis passwd. Enumere las diferencias encontradas en el resultado de cada uno de los comandos.

3. TECLADO / TERMINALES

3.1. ¿Qué sucede si tecleo cat /e <tab> p <tab>? (donde tab es la tecla tabulación). Presione <tab> nuevamente ¿Qué pasó ahora?
3.2. ¿Qué sucede si tecleo cat /e <tab> pas <tab>?
3.3. En este punto analizaremos las distintas terminales que hay en un sistema GNU/Linux. Ejecute los siguientes comandos e indique cuál fue el resultado:

3.3.1. who
3.3.2. Presione la tecla <alt>, y sin soltarla presione cualquiera de las teclas de función. En la pantalla debería aparecer el login del sistema, de lo contrario, ejecute el paso nuevamente presionando otra tecla de función. Si ya tiene el login del sistema vuelva a conectarse.
3.3.3. Ejecute nuevamente el comando who. ¿Qué diferencias encuentra con la primera vez que lo ejecutó?
3.3.4. Ejecute el comando who am i ¿qué muestra?, ¿Qué diferencias tiene con el comando ejecutado en el punto anterior?
3.3.5. Repita el paso 3.3.2 y el 3.3.3 hasta que no encuentre ninguna sesión para abrir.
3.3.6. Una vez terminado el punto anterior, Ud. se encontrará sesionado en el sistema como mínimo seis veces. Lo que acaba de hacer es abrir seis terminales virtuales (que podrían ser usadas por distintos usuarios, con diferentes perfiles), en la misma máquina. Así como existen terminales virtuales dentro del mismo equipo, si Ud. cuenta con una red, o con terminales tipo serie, podría abrir tantas sesiones de trabajo como Ud. quiera o necesite. Investigue e indique cómo se denominan los distintos tipos de terminales, y cuáles son los archivos que las representan (tip: busque en el directorio /dev).

4. DIRECTORIOS

4.1. ¿Para qué se usa el comando cd?. Ejecute las siguientes variantes de cd e indique cuál fue el resultado obtenido:
- 4.1.1. cd /
- 4.1.2. cd
- 4.1.3. cd /etc
- 4.1.4. cd..
- 4.1.5. cd ..

4.2. Bash sobre directorios:
- 4.2.1. ¿Cuál/es son las diferencias entre el path absoluto y el path relativo?
- 4.2.2. ¿Qué es lo que realizan las siguientes operaciones? (tip: si no encuentra la diferencia primero haga cd /, y luego vuelva a intentar)
  - 4.2.2.1. cd ~
  - 4.2.2.2. cd -
- 4.2.3. ¿Cuál es la diferencia entre cd .. y cd ~-?

4.3. Operaciones con directorios:
- 4.3.1. ¿Con qué comando se puede crear un directorio?.
- 4.3.2. ¿Con qué comando se puede borrar un directorio?.
- 4.3.3. ¿Qué sucede sí el directorio no está vacío?.
- 4.3.4. ¿Cómo puedo salvar la situación anterior? (Sin borrar uno a uno los archivos existentes).

4.4. ¿Qué significa la expresión ./ cuando se utiliza delante de un archivo? ¿Para qué sirve?

4.5. ¿Cómo puede moverse entre directorios sin utilizar el PATH completo?

4.6. ¿Cuál es el contenido de los siguientes directorios que confirman la estructura de cualquier sistema operativo GNU/Linux:?
- 4.6.1. /boot
- 4.6.2. /dev
- 4.6.3. /bin
- 4.6.4. /etc
- 4.6.5. /usr
- 4.6.6. /sbin
- 4.6.7. /root
- 4.6.8. /etc/rc.d (y todos los que están adentro)
- 4.6.9. /proc
- 4.6.10. /mnt
- 4.6.11. /usr/bin
- 4.6.12. /usr/sbin
- 4.6.13. /var
- 4.6.14. /usr/man (y todos los que están adentro)
- 4.6.15. /opt
- 4.6.16. /tmp

5. ARCHIVOS

5.1. ¿Qué hacen los siguientes comandos?
- 5.1.1. cp
- 5.1.2. mv
- 5.1.3. rm
- 5.1.4. rcp
- 5.1.5. rsh
- 5.1.6. scp
- 5.1.7. ssh

5.2. Para cada comando del punto anterior realice un ejemplo, e indique qué realizó.

5.3. ¿Con qué comando puedo concatenar el contenido de dos archivos?.
- 5.3.1. ¿Se puede usar ese comando para otra cosa?.

5.4. Haga un ls –l /dev
- 5.4.1. ¿Qué significa el primer carácter?
- 5.4.2. ¿Cuáles son todos los posibles valores que puede contener ese campo y que significa cada uno?

5.5. ¿Para qué sirve el comando touch? ¿qué utilidad le encuentra?

6. PERMISOS

6.1. Teniendo en cuenta el ls –l anterior, ¿indique que son los siguientes 9 caracteres? (sin considerar el primero sobre el que ya respondió anteriormente)

6.2. ¿qué significa cada caracter? ¿cómo están agrupados?

6.3. ¿Cómo se asignan los permisos? (detalle los comandos).

6.4. ¿Qué son el owner, y el group de un archivo?. ¿Se pueden cambiar?.

6.5. Intente cambiar los permisos de un archivo perteneciente al root (sesionado como usuario). Explique qué sucedió.

6.6. Explique la forma de cambiar los permisos con valores en octal.

6.7. ¿Cuál es el significado de los permisos en los directorios (se debe indicar que indica una r, una w, y una x)?

7. FILTROS

7.1. ¿Cuál es la diferencia de los comandos more, less y cat?. De un ejemplo de cada uno.
7.2. ¿Cuál es la diferencia entre tail y head?.
7.3. ¿Para qué sirve el comando wc y que indican los parámetros –c –l –w? ¿Proponga ejemplos de uso?
7.4. ¿Qué es lo que realiza el comando uniq?.
7.5. ¿Qué es lo que realiza el comando grep?.
7.5.1. ¿Para qué sirve?
7.5.2. ¿Qué hace la siguiente línea?: grep root /etc/passwd
7.5.3. ¿Qué diferencias encuentra entre la ejecución de los siguientes comandos?:
7.5.3.1. grep r /etc/passwd
7.5.3.2. grep ^r /etc/passwd
7.5.3.3. grep r$ /etc/passwd

8. **VI**

8.1. Ejecute la siguiente instrucción: vi $HOME/prueba.txt ¿Qué sucedió?. Ahora ejecute todos los pasos detallados a continuación.
8.1.1. Escriba la siguiente frase: “Este es el archivo prueba.txt de <nombre y apellido>”
8.1.2. ¿Qué tuvo que hacer para poder escribir la frase?
8.1.3. Guarde el archivo, y salga del editor. ¿Qué comando utilizó?
8.1.4. Ingrese nuevamente al archivo.
8.1.5. Incorpore al inicio del archivo el siguiente párrafo (los acentos puede ser evitados):
“Sistemas Operativos
Comisión de los días <día de cursada>
Trabajo Práctico 1
Alumno: <su nombre aquí>
Matrícula: <su matrícula aquí>
Documento: <su documento aquí>”
8.1.6. Describa todos los pasos que tuvo que realizar.
8.1.7. Guarde el archivo y continúe la edición. ¿Qué comandos utilizó?
8.1.8. Borre la línea de “Matrícula”. Indique por lo menos dos formas de realizarlo.
8.1.9. Invierta el orden de las líneas “Comisión y TP”. No está permitido rescribirlas. ¿Qué comandos utilizó?
8.1.10. Ubíquese en la línea 2 (dos) del archivo. No está permitido usar las teclas del cursor, ni el mouse. ¿Qué comando utilizó?
8.1.11. Marque para copiar las líneas 2, 3, y 4 (todas juntas, no de a una a la vez). ¿Cómo lo realizó?
8.1.12. Ubíquese al final del archivo (sin usar las teclas del cursor), y pegue dos veces el contenido del buffer. ¿Qué comando usó?
8.1.13. Deshaga uno de los copiados. No está permitido borrar línea por línea, ni caracter a caracter. ¿Qué comando usó?
8.1.14. ¿Cómo busco la palabra “Documento”? ¿Cómo busco la segunda ocurrencia de una palabra?
8.1.15. ¿Cómo puedo reemplazar la palabra “Documento” por “Documento:” (sin borrar, o realizar el reemplazo a mano)’
8.1.16. Guarde el archivo y salga.
8.1.17. Ejecutar "vi buscar_reemplazar" e introducir el texto:

```
1/5/2003 ---------- listo
1/5/2004 ---------- listo
1/5/2005 ---------- listo
1/5/2006 ---------- listo
1/5/2007 ---------- listo
1/5/2008 ---------- listo
1/5/2009 ---------- listo
1/5/2010 ---------- listo
1/5/2011 ---------- listo
1/5/2012 ---------- listo
1/5/2013 ---------- listo
1/5/2014 ---------- No listo
```

8.1.18. Ejecutar ":%s/\/3\//Marzo/g ¿Que paso al ejecutar esto?
8.1.19. Si observa el resultado de lo anterior, el cambio fue erróneo, modifique la sentencia para que funcione correctamente.
8.1.20. Modifique la fecha para que en lugar del 1 sea el 15. Indique que comandos uso para realizarlo.
8.1.21. ¿Indique si existe alguna forma de hacer un buscar y reemplazar pero que antes de realizar la substitución pregunte?.

9. DISCO

9.1. ¿Para qué se utiliza el comando mount?. ¿Todos los usuarios lo pueden ejecutar el comando con algunos o todos los parámetros?. En caso de que su respuesta sea negativa, ¿indique cuál /es si?.

9.2. Transfiera el archivo a un disquete (el mismo que utilizará para entregar el trabajo práctico, ya que éste archivo es parte de la entrega).

9.2.1. Indique al menos dos formas de realizarlo.

9.3. ¿Recordó desmontar el disquete en todas las oportunidades que lo uso, y antes de retirarlo verdad?.

9.3.1. ¿Qué problemas se pueden generar por no realizarlo?.

9.3.2. Repita el punto 1 de éste trabajo práctico, creando un usuario cualquiera (si Ud. se encuentra en el Lab 266, no puede continuar con éste punto).

9.3.3. Cambie de terminal virtual a otra, si se encuentra sesionada salga, e ingrese con el usuario creado en el punto anterior.

9.3.4. Intente desmontar el disquete. ¿Pudo?. Si su respuesta es negativa lo mismo pasará en el laboratorio si Ud. se retira de trabajar sin desmontar la disquetera, y el próximo usuario la quiere utilizar. Por ese motivo en el laboratorio al hacer el logout del sistema se ejecuta un script que verifica si la disquetera está montada. En caso de estarlo, la desmonta, y además envía un alerta administrativo a los administradores de la red. Al tercer alerta administrativo que se genere se le bloqueará la cuenta por un período de 15 días.

9.3.5. ¿De qué manera nombra el sistema a cada unidad de disco?

9.3.6. ¿Cómo identifica Ud. a qué unidad se hace referencia?

9.3.7. ¿Podría Ud. indicar en que unidad y partición se encuentra instalado el GNU/Linux en su computadora? ¿Qué comandos o archivos de información utilizó?

10. VARIABLES DE ENTORNO

10.1. ¿Qué son las variables de entorno y para qué sirven?.

10.1.1. Escriba el contenido y explique el significado de las siguientes variables: HOME / LOGNAME / PATH / HOSTNAME / IFS

10.1.2. ¿Qué comando usó para ver el contenido de las variables del punto anterior?

10.1.3. Cree una variable de entorno HOLA que contenga el mensaje “Hola mundo”.

10.1.4. ¿Cuál es el uso que le da el sistema a la variable PATH? ¿Qué ocurres si intenta ejecutar un comando que no se encuentra ubicado en alguno de los directorios que contiene la variable? ¿Cómo lo soluciona?

10.1.5. ¿Por qué existen las variables PS1 y PS2? ¿Qué es un comando multilínea?

11. PLACA DE RED (En caso de no tener en su máquina, realizarlo en el Lab266)

11.1. ¿Para qué sirve el comando ifconfig y en que directorio se encuentra?

11.2. ¿Qué IP o IP’s tiene asignada la computadora?

11.3. ¿Qué es el adaptador lo y para que se utiliza?

11.4. ¿Cuál es la salida del comando ping -c4 (ip del eth0)?

Trabajo Práctico Nro. 1 (GRUPAL):

- ***Descripción:*** Se programarán todos los scripts mencionados en el presente trabajo, teniendo especialmente en cuenta las recomendaciones sobre programación mencionadas en la introducción de este trabajo. Los contenidos de este trabajo práctico pueden ser incluidos como tema de Parciales.
- ***Formato de entrega:*** Electrónico en el Laboratorio 266 siguiendo el protocolo especificado
- ***Fecha de entrega:*** No se aceptará una entrega posterior sin tener entregadas todas las anteriores.
- ***Documentación:*** Todos los scripts que se entreguen deben tener un encabezado y un fin de archivo. Dentro del encabezado deben figurar el nombre del script, el trabajo práctico al que pertenece y el número de ejercicio dentro del trabajo práctico al que corresponde, el nombre de cada uno de los integrantes detallando nombre y apellido y el número de DNI de cada uno (tenga en cuenta que para pasar la nota final del trabajo práctico será usada dicha información, y no se le asignará la nota a ningún alumno que no figure en todos los archivos con todos sus datos), también deberá indicar el número de entrega a la que corresponde (entrega, primera reentrega, segunda reentrega, etc.). Para su mejor comprensión del tema vea el ejercicio 4 de la primera entrega que fue diseñado para que su trabajo sea más fácil.
- ***Evaluación:*** Luego de entregado el trabajo práctico los ayudantes procederán a evaluar los ejercicios resueltos, en caso de encontrar errores se documentará en la carátula del TP que será devuelta al grupo con la evaluación final del TP y una fecha de reentrega en caso de ser necesaria (en caso de no cumplir con dicha fecha de reentrega el trabajo práctico será desaprobado). Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el trabajo práctico presentado.
  Las notas sobre los trabajos también estarán disponibles en el sitio de la cátedra (www.sisop.com.ar) donde además del estado de la corrección se podrá ver un detalle de las pruebas realizadas y los defectos encontrados

  Importante: Cada ejercicio cuenta con una lista de validaciones mínimas que se realizará, esto no implica que se puedan hacer otras validaciones al momento de evaluar el trabajo presentado

Introducción:

La finalidad del presente práctico es que los alumnos adquieran un cierto entrenamiento sobre la programación de shell scripts, practicando el uso de utilitarios comunes. Todos los scripts, están orientados a la administración de una máquina, o red GNU/Linux y pueden ser interrelacionados de tal manera de darles una funcionalidad real.
Cabe destacar que todos los scripts deben poder ser ejecutados en forma batch o interactiva, y que en ningún caso serán probados con el usuario root, por lo cual deben tener en cuenta al realizarlos, no intentar utilizar comandos, directorios, u otros recursos que solo están disponibles para dicho usuario, o usuarios del grupo. Todos los scripts deberán funcionar en las instalaciones del laboratorio 266, ya que es ahí donde serán controlados.
Para un correcto y uniforme funcionamiento, todos ellos deberán respetar algunos lineamientos generales:

1. Modularidad:

Si bien es algo subjetivo del programador, se trata de privilegiar la utilización de funciones (internas / externas), para lograr una integración posterior menos trabajosa.

2. Claridad:

Se recomienda fuertemente el uso de comentarios que permitan la máxima legibilidad posible de los scripts.

3. Verificación:

Todos los scripts deben realizar un control de las opciones que se le indiquen por línea de comando, es decir, verificar la sintaxis de la misma. En caso de error u omisión de opciones, al estilo de la mayoría de los comandos deben indicar mensajes como:

```
"Error en llamada!
 Uso: comando [...]"
```

Donde se indicará entre corchetes [ ], los parámetros opcionales; y sin ellos los obligatorios, dando una breve explicación de cada uno de ellos.
Todos los scripts deben incluir una opción standard '-?' que indique el número de versión y las formas de llamada.

Ejercicios:

- ***Fecha de entrega:*** Del **15/09/2014 al 19/09/2014,** dependiendo del día que se curse la materia y sólo en la primer fecha en que se cursa (lunes / martes / viernes).
- ***Fecha tardía de entrega:*** Del **22/09/2014 al 26/09/2014,** dependiendo del día que se curse la materia y sólo en la primer fecha en que se cursa (lunes / martes / viernes).

Los ejercicios que se encuentran dentro de esta sección son básicos, y fueron diseñados para que los alumnos adquieran un primer contacto con los comandos de GNU/Linux, y con la forma de programar scripts, incluyendo awk.

Tip: En caso de tener problemas con un script bajado desde un disquete formateado con DOS, y que contiene ^M al final de cada línea puede usar el siguiente comando para eliminarlos:

```
tr -d '\r' <archivo_con_M >archivo_sin_M
```

En caso de ejecutar scripts que usen el archivo de passwords, en el laboratorio, debe cambiar "cat /etc/passwd" por "getent passwd"

Ejercicio 1:

Dado el siguiente script:

```
#!/bin/bash

if test $# -lt 2
then
    echo "..."
    exit 1;
fi

if  ! test -f $1
then
    echo "..."
    exit 1;
fi

TOT=$#
((TOT-=2))
FILE=$1
shift 1
COUNTER=0
while test $COUNTER -le $TOT
do
    SET[$COUNTER]=$1
    shift 1
    ((COUNTER++))
done

for X in "${SET[@]}"
do
    CANT=0
    for WORD in `cat $FILE`
    do
            if [[ "$WORD" == "$X" ]]
            then
                    ((CANT++))
            fi
    done
    echo "$X : $CANT"
done

exit 0
```

a) Analice el código y complete las líneas que tienen puntos suspensivos.
b) El script necesita parámetros. ¿Cuáles son?
c) Analice los resultados al ejecutar el script, ¿Que significa lo que se muestra por pantalla?.
d) Explique el objetivo general.
e) Investigue y redacte en pocas líneas sobre cada uno de los comandos: shift, test y cat.
f) Explique cómo se utilizan en bash las sentencias if, for y while.

Ejercicio 2:

Se pide generar un script que ordene una serie de números enteros. Para esto el script deberá recibir como parámetro el sentido en que se quiere ordenar, siendo "**-a**" para ascendente y "**-d**" para descendente; seguido de la serie de números a ordenar. Por ejemplo, la siguiente llamada al script:

./script.sh -a 10 5 4 7

debería devolver los números ordenados ascendentemente: *4 5 7 10*

Consideraciones:

- El parámetro del orden (**-a** o **-d**) siempre deberá ser el primer parámetro.
- Los números pueden ser tanto negativos como positivos, pero siempre enteros.

Criterios de corrección:

Control	Criticidad
Valida correctamente todos los parámetros recibidos en cuanto a cantidad y tipo.	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Debe manejar correctamente más de 10 parámetros	Obligatorio

Ejercicio 3:

Se pide que genere un script que reciba por parámetro el nombre de un archivo de texto, el cual tendrá dentro del mismo varias líneas repetidas. El script deberá tomar dicho archivo e informar por pantalla cuantas veces aparece cada línea dentro del mismo. No debe hacerse distinciones entre mayúsculas y minúsculas; es decir que las líneas "linea 1", "Linea 1" y "LINEA 1" deben considerarse como una misma línea repetida varias veces en el archivo.

Por ejemplo, teniendo un archivo con el contenido de la izquierda el script debería informar algo similar al cuadro de la derecha:

Marcelo Agustin Damian marcelo Natalia MARCELO DamiAn	agustin (1) damian (2) marcelo (3) natalia (1)

Criterios de corrección:

Control	Criticidad
Valida todos los parámetros recibidos en cantidad y tipo	Obligatorio
Implementado solo en bash (no se acepta AWK)	Obligatorio
El archivo original no debe sufrir ninguna modificación	Obligatorio
No hace diferencia en entre mayúsculas y minúsculas	Obligatorio
En caso de crear archivos temporales estos deben ir en el directorio /tmp y deben ser eliminados por el script al finalizar.	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Comentarios en el código según sea necesario	Deseable

Ejercicio 4:

Se cuenta con un archivo usuarios de un laboratorio, con el mismo formato del archivo de passwords (/etc/passwd), más un campo al final del registro conteniendo fecha de creación del usuario.
El home de los usuarios, se organiza de la siguiente manera:

/home/alumnos/comision
/home/docentes

El proceso que genera este archivo, fue modificado en distintas oportunidades, y se modificó el formato con que se guarda la fecha de creación, por lo que algunos registros tienen el formato **dd-MM-yyyy hh:mm:SS**, mientras que otros tienen el formato **dd/MM/yy** y otros **dd de MM de yyyy**.

Se pide realizar un script con **AWK** que reciba por parámetro el nombre del archivo de usuarios y opcionalmente una fecha. El objetivo del script es informar los alumnos que fueron creados en esa fecha, ordenados por comisión. Si no se especificó una fecha por parámetro, se debe considerar la fecha de hoy.

Consideraciones:

- El formato del parámetro **fecha** puede ser **dd/MM/yyyy** o **dd/MM/yy.**

- La forma de invocación debe ser:
  ./ejercicio4 <archivo de usuarios> [fecha]
- El formato de salida debe ser:
  Usuarios creados el <fecha>

Comision	**Apellido y nombre**	**Usuario**
lumi	**Perez Jose**	**joseperez**
maju	**Sara Juana**	**sarajuana**
maju	**Lopez Juan**	**juanlopez**

Criterios de corrección:

Control	Criticidad
El filtro de registros debe hacerse utilizando expresiones regulares	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Se debe proveer un archivo de usuarios de ejemplo junto con la entrega del script	Obligatorio
Se debe validar que el archivo recibido cumpla con el formato esperado	Deseable
Si no existen usuarios que cumplan con el criterio de búsqueda, se debe informar	Obligatorio
Se validarán los parámetros recibidos, tanto en formato, cantidad y ubicación	Obligatorio

Ejercicio 5:

Se cuenta con un archivo que contiene las notas del año de los alumnos de una materia, con el siguiente formato:

parcial_nro, alumno_nombre, alumno_dni, nota

Dentro de este archivo sólo existirán registros para los parciales rendidos y no se anotarán los ausente. La cantidad de parciales a rendir por cada alumno es dos. Cualquier alumno que posea menos de dos parciales rendidos tendrá la condición de desaprobado directamente y si por error existiesen más de dos parciales para el mismo alumno se deberá informar la situación como error.

Ejemplo:
$cat Parciales_2014_Q2.txt
1, Hernán Hernandez, 30330333, 5
1, Fernando Fernandez, 34340344, 4
2, Hernán Hernandez, 30330333, 6
2, Fernando Fernandez, 34340344, 2
1, Juan Hernandez, 30330355, 7

Se pide confeccionar un script usando *awk* para generar reportes en base al archivo de entrada, de acuerdo a estos parámetros:

-p → Reporte de parciales. Muestra el número total de alumnos y un reporte por cada parcial incluyendo:
- Cantidad de alumnos que sacaron nota igual a 7 o más
- Cantidad de alumnos que sacaron entre 4 y 6
- Cantidad de desaprobados
- Muestra el promedio de notas por parcial

Ejemplo:
./notas Parciales_2014_Q2.txt -p
Alumnos: 3
Parcial1: Promoción: 1, Entre 4 y 6: 2, Desaprobados: 0, Promedio: 5.3
Parcial2: Promoción: 0, Entre 4 y 6: 1, Desaprobados: 1, Promedio: 4

-a → Reporte de alumnos. Muestra para cada alumno el promedio y condición (materia aprobada, cursada o desaprobada).

Ejemplo:
./notas Parciales_2014_Q2.txt -a
Hernán Hernández, DNI: 30330333, Materia cursada
Fernado Fernandez, DNI: 34340344, Materia desaprobada
Juan Hernandez, DNI: 30330355, Materia desaprobada

-a alumno_dni → Consulta de alumno. Muestra las notas de cada parcial del alumno indicado, su nombre, dni, el promedio y condición (materia aprobada, cursada o desaprobada).

Ejemplo:
./notas Parciales_2014_Q2.txt –a 30330333
Alumno: Hernán Hernández, DNI: 30330333
Parcial Nota
1 5
2 6
Promedio: 5.5 Materia cursada.

Criterios de corrección:

Control	Criticidad
Validación de parámetros: el script recibe como parámetro el archivo a procesar (requerido), más la opción del reporte a realizar (requerido), también se debe incluir la opción de ayuda -?. Se debe validar cantidad y tipo de parámetros.	Obligatorio
Formato de salida: Respetar el formato del ejemplo	Obligatorio
Funcionamiento: cumple con los parámetros solicitados y muestra el contenido del resultado en pantalla.	Obligatorio
El script ofrece ayuda con –h, -? o –help explicando cómo se lo debe invocar	Obligatorio
Presentación: se debe presentar un archivo de entrada de ejemplo junto con los entregables	Obligatorio
Comentarios en el código	Obligatorio
Uso de arrays asociativos	Obligatorio

Ejercicio 6 (entrega tardía):

Genere un script que reciba por parámetro un archivo de texto con información de los goleadores del torneo Inicial de fútbol de Primera Nacional, cada línea del archivo de texto recibido contendrá la fecha, nombre de jugador y cantidad de goles. El script debe sumar los goles de cada fecha por cada jugador y generar en un archivo de salida ordenado de mayor a menor según la cantidad de goles de cada jugador. Si dos jugadores tienen igual cantidad de goles entonces se ordenarán en segundo orden alfabéticamente.

Por ejemplo:

Entrada.txt

```
Fecha          Jugador                    Total
4/9/2014       Scocco, Ignacio            2
5/9/2014       Gigliotti, Emanuel         1
5/9/2014       Figueroa, Victor           1
5/9/2014       Nanni, Roberto             1
5/9/2014       Heinze, Gabriel            1
8/9/2014       Gigliotti, Emanuel         2
8/9/2014       Figueroa, Victor           1
12/9/2014      Zapata, Duván Estevan      1
12/9/2014      Rescaldani, Ezequiel       1
13/9/2014      Riaño, Claudio             2
16/9/2014      Zapata, Duván Estevan      1
16/9/2014      González, Federico Rafael  2
16/9/2014      Zapata, Duván Estevan      1
16/9/2014      Penco, Sebastián Ariel     1
```

Salida.txt

```
Jugador                    Total
Gigliotti, Emanuel         3
Zapata, Duván Estevan      3
Figueroa, Victor           2
González, Federico Rafael  2
Riaño, Claudio             2
Scocco, Ignacio            2
Heinze, Gabriel            1
Nanni, Roberto             1
Penco, Sebastián Ariel     1
Rescaldani, Ezequiel       1
```

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Se debe proveer un archivo de ejemplo junto con la entrega del script	Obligatorio
El script debe validar el formato del archivo de entrada	Obligatorio
El script ofrece ayuda con –h, -? o –help explicando cómo se lo debe invocar	Obligatorio
Se debe poder parametrizar la ruta y nombre del archivo de salida.	Obligatorio
Comentarios en el código	Obligatorio
Uso de arrays asociativos	Deseable

Trabajo Práctico Nro. 2 (GRUPAL):

- ***Descripción:*** Codificar todos los programas mencionados en el presente trabajo, teniendo especialmente en cuenta las recomendaciones sobre programación mencionadas en la introducción de este trabajo. Los contenidos de este trabajo práctico pueden ser incluidos como tema de Parciales. Para la entrega de éste trabajo práctico se deberán tener en cuenta los siguientes puntos:

  a) Todos los ejercicios deben tener incorporado manejo de proyectos, lo que implica la generación de bibliotecas (.h), para la compilación se deben generar *makefile's*, y en caso de necesitar archivos de parametrización los mismos deben estar en formato unix.
  b) Código autodocumentable. Esto quiere decir que cada función (u otra estructura), debe contener un comentario que indica claramente para que se usa, cual es el propósito, etc. Asimismo dentro del código de la función debe tener comentarios de lo que se está intentando hacer en cada momento.
  c) Manual de configuración. Cualquier proyecto que le entregue a la cátedra deberá tener un documento (impreso o en el mismo directorio), conteniendo la forma de utilizar y parametrizar el programa.

- ***Formato de entrega:*** ver reglamento de entregas
- ***Fecha de entrega:***. No se podrá aceptará una entrega posterior sin tener entregadas todas las anteriores.
- ***Evaluación:*** Luego de entregado el trabajo práctico los ayudantes procederán a evaluar los ejercicios resueltos, en caso de encontrar errores se documentará en la carátula del TP que será devuelta al grupo con la evaluación final del TP y una fecha de reentrega en caso de ser necesaria (en caso de no cumplir con dicha fecha de reentrega el trabajo práctico será desaprobado). Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el trabajo práctico presentado. Es importante indicar que dentro de la evaluación se tendrá en cuenta si se cumple o no con las peticiones de tener makefiles, la buena codificación y documentación del código fuente.
  Las notas sobre los trabajos también estarán disponibles en el sitio de la cátedra (www.sisop.com.ar) donde además del estado de la corrección se podrá ver un detalle de las pruebas realizadas y los defectos encontrados

  Importante: Cada ejercicio cuenta con una lista de validaciones mínimas que se realizará, esto no implica que se puedan hacer otras validaciones al momento de evaluar el trabajo presentado

Ejercicios: fork, procesos concurrentes, zombies, Exec, señales y FIFOS:

- ***Fecha de entrega:*** Del **06/10/2014 al 10/10/2014,** dependiendo del día que se curse la materia y sólo en la primer fecha en que se cursa (lunes / martes / viernes).
- ***Fecha tardía de entrega:*** Del **13/10/2014 al 17/10/2014,** dependiendo del día que se curse la materia y sólo en la primer fecha en que se cursa (lunes / martes / viernes).

Ejercicio 1:

Para cada uno de los siguientes programas:

1) Codifique, compile con gcc y ejecute pasando como parámetro el número **5**.
2) Evalúe los resultados impresos por pantalla:
   a. ¿Cuántos procesos se crean al ejecutar el programa?
   b. Realice un árbol donde se visualice la relación entre dichos procesos.
3) Investigue el comportamiento de *fork()* y explique brevemente como debe utilizarse.

Programa A:

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>

int errorMessage(const char *msg);

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

```
        if(argc<2)
           return errorMessage("Debe ingresar parametros");

        if(atoi(argv[1])<0)
           return errorMessage("Error en la cantidad de procesos a crear");

        for(int x=0;x<atoi(argv[1]);x++) {
           pid = fork();
           if (pid==-1)
              return errorMessage("Error en la creaciÃ³n del proceso");
           else
              if (pid==0) {
                 printf("soy el proceso hijo con pid %d\n", getpid());
                  return 0;
              }
              else
                 printf("Soy el proceso padre %d con padre %d \n",getpid(), getppid());
        }

        for(int x=0;x<atoi(argv[1]);x++)
           wait(NULL);
}

int errorMessage(const char *msg) {
   printf("%s\n",msg);
   return 1;
}
```

Programa B:

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>

int errorMessage(const char *msg);

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

   if(argc<2)
      return errorMessage("Debe ingresar parametros");

   if(atoi(argv[1])<0)
      return errorMessage("Error en la cantidad de procesos a crear");

   for(int x=0;x<atoi(argv[1]);x++) {
      pid = fork();
      if (pid==-1)
         return errorMessage("Error en la creaciÃ³n del proceso");
      else
         if (pid==0) {
            printf("soy el proceso hijo con pid %d\n", getpid());
            sleep(1);
         }
         else {
            printf("Soy el proceso padre %d con padre %d \n",getpid(), getppid());
            break;
         }
   }

   for(int x=0;x<atoi(argv[1]);x++)
      wait(NULL);
}

int errorMessage(const char *msg) {
   printf("%s\n",msg);
   return 1;
}
```

Ejercicio 2:

Realice un programa que por línea de comandos reciba un parámetro g que específica una cantidad de generaciones de hijos a ser creadas. Cada hijo creado deberá:

- Crear dos hijos.
- Mostrar por pantalla: pid, parentesco con el padre del árbol de procesos, pid del padre y pids de los hijos.

Al finalizar la ejecución de todas las generaciones de hijos el padre deberá informar por pantalla su finalización y salir.

Ejemplo de salida:

Soy 121 hijo de 120, padre de 122 y 123
Soy 122 hijo de 121, nieto de 120, padre de 124 y 125
Soy 123 hijo de 121, nieto de 120, padre de 126 y 127
Soy 124 hijo de 122, bisnieto de 120, padre de 128 y 129
…
Soy 1622 hijo de 1621, tatara-tatara-tatara-...-tataranieto de 120, sin hijos..
Toda mi familia terminó – Fin de la ejecución.

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
No quedan procesos zombies y maneja correctamente finalización normal y abrupta	Obligatorio
Validación de parámetros de entrada	Obligatorio
Código debidamente comentado	Deseable

Ejercicio 3:

Se pide generar un proceso para medir los tiempos de ejecución de otros procesos. Para esto, crear un proceso padre que genere un número variable de procesos hijos, lo cual será determinado por parámetro, y lleve la cuenta de los segundos de ejecución de cada proceso.

Los procesos hijos llevaran el conteo de los segundos de ejecución y finalizarán al recibir la señal SIGUSR1, el resto de las señales deben ser ignoradas. Al finalizar cada proceso debe informar el tiempo de ejecución y el proceso principal debe informar el PID del proceso que finalizó y también debe informar el tiempo de ejecución del proceso que acaba de finalizar, de manera de poder corroborar que se llevó la misma cuenta en ambos procesos.

El tiempo de ejecución de los procesos hijos deberá ser aleatorio entre 3 y 10 segundos. No se permite la utilización de la sentencia **sleep** ya que se pretende que el hijo consuma ciclos de CPU.

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
No quedan procesos zombies y maneja correctamente finalización normal y abrupta	Obligatorio
Debe validar correctamente los parámetros de entrada y disponer de ayuda (-h o -?).	Obligatorio
No se utiliza el comando “sleep” o derivados	Obligatorio
Código debidamente comentado	Deseable

Ejercicio 4:

Para el monitoreo de largos procesos que se ejecutan al cierre del día, se necesita un comando que permita ver el resultado por pantalla, para detectar si hubo errores, pero además almacene la salida en un archivo de log, para dejar registro diario de los procesos.
Se debe crear un programa que ejecute un comando pasado por parámetro, y tomando la salida que genere dicho comando lo muestre por pantalla en forma paginada (como hacen less o more) y además guarde la salida en el archivo de log, poniendo como encabezado el comando ejecutado, y la fecha y hora de inicio; y al finalizar el comando agregue al log una línea indicando la finalización y la fecha y hora.

Deberá aceptar como primer parámetro el archivo de log a guardar, el siguiente parámetro el comando a ejecutar y a continuación los parámetros para ejecutar dicho comando (si los necesita).

Importante: No se evaluará este ejercicio con comandos que requieran el ingreso de datos en forma interactiva (ej. more, top, ftp, etc)

Ejemplo de llamada:

```
$ ./paginador archivo.log ls -l
```

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
Permite pasar parámetros al comando a ejecutar	Obligatorio
Validación de parámetros de entrada	Obligatorio
No genera archivos temporales	Obligatorio
Permite generar el archivo de log en cualquier directorio	Obligatorio
Maneja correctamente la salida de errores (stderr)	Obligatorio
Código debidamente comentado	Deseable

Ejercicio 5:

Una empresa desea mantener en un equipo servicios que deben estar en ejecución las 24 horas.
Solicitan la realización de un programa demonio que, leyendo una lista de aplicaciones (con sus respectivos parámetros de línea de comandos), las ejecute y en caso de que alguna se caiga, las reinicie.

El demonio solo debe terminar ante una señal SIGTERM.
Antes de finalizar la ejecución deberá detener todos los servicios en ejecución, para ello primero intentará enviar una señal SIGTERM y si pasados 10 segundos la aplicación continúa en ejecución, se terminará con SIGKILL.

Se requiere, además, que el programa genere un archivo de log para cada aplicación con el output de la misma en tiempo real y un archivo de log general donde se registren los siguientes eventos:

- Inicio del demonio
- Ejecución de una aplicación
- Muerte de una aplicación
- Reinicio de una aplicación
- Detención de una aplicación.
- Detención del demonio

En todos los casos indique fecha y hora y pid del proceso
En caso de la detención de una aplicación indique también la señal enviada (SIGTERM / SIGKILL)

Ejemplo de ejecucion**:**
./restarter servicios.db

Contenido de ejemplo de servicios.db**:**
firefox
ping 192.168.1.1
/usr/bin/programa

Salida:

log/ping.log

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=159 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=3.54 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=3.25 ms
64 bytes from 192.168.1.1: icmp_req=4 ttl=64 time=3.15 ms
64 bytes from 192.168.1.1: icmp_req=5 ttl=64 time=3.20 ms

log/restarter.log

[17/08/2014 - 19:25] Iniciando Restarter
[17/08/2014 - 19:25] Ejecutando firefox (pid 7162)
[17/08/2014 - 19:25] Ejecutando ping (pid 7164)
[17/08/2014 - 19:25] Ejecutando /usr/bin/programa (pid 7172)
[17/08/2014 - 20:11] Se cerró firefox (pid 7162)
[17/08/2014 - 20:11] Reiniciando firefox (pid 8523)
[17/08/2014 - 20:21] Deteniendo firefox (SIGTERM) (pid 8523)
[17/08/2014 - 20:21] Deteniendo ping (SIGTERM) (pid 7164)
[17/08/2014 - 20:21] Deteniendo /usr/bin/programa (SIGTERM) (pid 7172)
[17/08/2014 - 20:21] Deteniendo firefox (SIGKILL) (pid 8523)
[17/08/2014 - 20:21] Deteniendo restarter

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
Resolver utilizando funciones de la familia exec.	Obligatorio
Validación de parámetros de entrada, ayuda en línea y archivo de entrada.	Obligatorio
No genera procesos zombies y maneja correctamente la finalización normal y abrupta	Obligatorio
Maneja correctamente las señales	Obligatorio
Código debidamente comentado	Deseable

Ejercicio 6 (entrega tardía):

Diseñe e implemente un sistema que mida la aleatoriedad en la generación de números. Para ello, cree 10 procesos que deberán generar 100 números enteros aleatorios cada uno, entre 0 y 9. Cada proceso, a medida que genera un número aleatorio se lo comunica a un concentrador de números (otro proceso) a través de un FIFO y descansa un tiempo aleatorio no mayor a medio segundo.

El proceso concentrador será el encargado de recibir los números aleatorios de todos los procesos generadores, y llevar una contabilidad de la cantidad de ocurrencias de cada valor recibido. Al finalizar, el concentrador deberá mostrar una tabla con la contabilidad elaborada.

Criterios de corrección:

Control	Criticidad
Permite utilizar una semilla ingresada por parámetro	Obligatorio
En caso de no recibir una semilla, la genera automáticamente	Obligatorio
No deja recursos bloqueados del sistema al finalizar la ejecución	Obligatorio
Los procesos generadores y concentradores no deben ser parientes.	Obligatorio
Brinda información estadística sobre los resultados obtenidos en la contabilidad	Deseable
Código debidamente comentado	Obligatorio

Trabajo Práctico Nro. 3 (GRUPAL):

- ***Descripción:*** Codificar todos los programas mencionados en el presente trabajo, teniendo especialmente en cuenta las recomendaciones sobre programación mencionadas en la introducción de este trabajo. Los contenidos de este trabajo práctico pueden ser incluidos como tema de Parciales. Para la entrega de éste trabajo práctico se deberán tener en cuenta los siguientes puntos:

    a) Todos los ejercicios deben tener incorporado manejo de proyectos, lo que implica la generación de bibliotecas (.h), para la compilación se deben generar *makefile's*, y en caso de necesitar archivos de parametrización los mismos deben estar en formato unix.
    b) Código autodocumentable. Esto quiere decir que cada función (u otra estructura), debe contener un comentario que indica claramente para que se usa, cual es el propósito, etc. Asimismo dentro del código de la función debe tener comentarios de lo que se está intentando hacer en cada momento.
    c) Manual de configuración. Cualquier proyecto que le entregue a la cátedra deberá tener un documento (impreso o en el mismo directorio), conteniendo la forma de utilizar y parametrizar el programa.

- ***Formato de entrega:*** ver reglamento de entregas
- ***Fecha de entrega:*** No se aceptará una entrega posterior sin tener entregadas todas las anteriores.
- ***Evaluación:*** Luego de entregado el trabajo práctico los ayudantes procederán a evaluar los ejercicios resueltos, en caso de encontrar errores se documentará en la carátula del TP que será devuelta al grupo con la evaluación final del TP y una fecha de reentrega en caso de ser necesaria (en caso de no cumplir con dicha fecha de reentrega el trabajo práctico será desaprobado). Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el trabajo práctico presentado. Es importante indicar que dentro de la evaluación se tendrá en cuenta si se cumple o no con las peticiones de tener makefiles, la buena codificación y documentación del código fuente.
  Las notas sobre los trabajos también estarán disponibles en el sitio de la cátedra (www.sisop.com.ar) donde además del estado de la corrección se podrá ver un detalle de las pruebas realizadas y los defectos encontrados

Importante: Cada ejercicio cuenta con una lista de validaciones mínimas que se realizará, esto no implica que se puedan hacer otras validaciones al momento de evaluar el trabajo presentado

Ejercicios: uso de semáforos, memoria compartida, sockets y Threads:

- ***Fecha de entrega:*** Del **27/10/2014 al 31/10/2014,** dependiendo del día que se curse la materia y sólo en la primer fecha en que se cursa (lunes / martes / viernes).
- ***Fecha tardía de entrega:*** Del **03/11/2014 al 07/11/2014,** dependiendo del día que se curse la materia y sólo en la primer fecha en que se cursa (lunes / martes / viernes).

Ejercicio 1:

Se deben crear tres procesos. El primero debe calcular una x cantidad de senos matemáticos, el segundo la misma cantidad de cosenos y el tercero la misma cantidad de tangentes. El número de funciones trigonométricas a calcular debe ingresada por parámetro al inicio del ejercicio. El valor del parámetro de cada una debe ser un número aleatorio (es decir, el valor sobre el que se calculará la función trigonométrica). Cada vez que cada proceso termina de realizar los cálculos de todas las funciones debe ingresar en un archivo común a los tres procesos un mensaje diciendo "Soy el proceso 1, 2 ó 3 y calculé X senos/cosenos/tangentes en tantos segundos/milisegundos (lo que corresponda)" además de mostrar por pantalla el mismo mensaje. Algunas salidas de ejemplo podría ser:

Soy el proceso 2 y calculé 100000 cosenos en 3 segundos
Soy el proceso 1 y calculé 100000 senos en 2,5 segundos
Soy el proceso 1 y calculé 100000 senos en 2,3 segundos
Soy el proceso 3 y calculé 100000 tangentes en 2,2 segundos
y así sucesivamente...

Cada proceso debe realizar esta operatoria 10 veces y quedarse esperando que los demás terminen. Cuando todos los procesos terminen de ejecutar los 10 ciclos, el proceso 1 (el que calcula los senos) debe comenzar nuevamente otros 10 ciclos pero esta vez, la alternancia entre los tres tiene que ser estricta, es decir, primero debe ejecutar el primer ciclo de cálculo el proceso 1, luego el primer ciclo de

cálculo el proceso 2, luego el primer ciclo de cálculo el proceso 3, luego el segundo ciclo de cálculo el proceso 1 y así sucesivamente. Las salidas en este caso debería ser algo así como:

Soy el proceso 1 y calculé 100000 senos en 2,8 segundos
Soy el proceso 2 y calculé 100000 cosenos en 1,5 segundos
Soy el proceso 3 y calculé 100000 tangentes en 2,2 segundos
Soy el proceso 1 y calculé 100000 senos en 2,5 segundos
Soy el proceso 2 y calculé 100000 cosenos en 2,5 segundos
y así sucesivamente...

Cuando el proceso 3 termine su último ciclo termina el ejercicio.

Nota: Recordar que la tangente de 90 grados no está definida.

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado (correcta sincronización)	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
Finaliza correctamente y no deja procesos zombies o huérfanos	Obligatorio
Utiliza y elimina correctamente los semáforos definidos	Obligatorio
Maneja correctamente la finalización normal y abrupta (kill)	Obligatorio
Código debidamente comentado	Deseable

Ejercicio 2:

Se tiene un archivo de texto con cualquier contenido.
Se tiene, además, un primer proceso que va leyendo el contenido del archivo y se lo tiene que ir pasando por memoria compartida a un segundo proceso.
El segundo proceso debe ir recuperando la información de la memoria compartida y tiene que ir mostrando su contenido por pantalla. Al terminar el primer proceso de leer todo el archivo y el segundo recuperar la última información de la memoria compartida se debería tener en pantalla el contenido del archivo de texto (mostrado por el segundo proceso).
El método de traspaso de información a través de la memoria compartida debe ser el siguiente. La primera vez el primer proceso lee un solo caracter del archivo y espera a que el segundo lo lea. Luego el primer proceso lee dos caracteres y espera a que el segundo los lea, y así sucesivamente leyendo cada vez un caracter más hasta llegar a un tope de 15 caracteres leídos de una sola vez. Al llegar a 15 se irá descendiendo en la totalidad de caracteres leídos de una sola vez de uno en uno, es decir, primero 14, luego 13, etc. Si se llegase nuevamente a 1 antes de terminar de leer el archivo de texto otra vez hay que ir incrementando la cantidad de caracteres leídos de una sola vez de uno en uno y así hasta que se termine con el contenido del archivo.
Para poder ir viendo por pantalla este comportamiento asegurarse de poner un delay de no menos de 500 ms (milisegundos) entre leída y leída.

Criterios de corrección:

Control	Criticidad
El nombre del archivo a leer debe ser pasado por parámetro	Obligatorio
Valida correctamente la existencia y tipo de archivo recibido por parámetro	Obligatorio
No quedan recursos tomados después de la ejecución del programa	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Código debidamente comentado	Obligatorio

Ejercicio 3:

Se requiere confeccionar un programa que monitoree un directorio y procese los archivos que se encuentran en el mismo, a fin de informar todos los caracteres distintos que contenga el archivo, y la cantidad de cada uno.
El programa debe recibir como parámetros el directorio a monitorear, y la cantidad de hilos a utilizar (obligatorios, en ese orden). Al iniciar, debe procesar todos los archivos que se encuentran en el directorio (uno por hilo) o que se agreguen al mismo mientras está en ejecución. En caso de que el directorio quede vacío, el programa deberá realizar una espera pasiva hasta que se agreguen nuevos archivos, o bien sea finalizado.
Cada archivo deberá ser procesado por un hilo distinto, hasta un máximo de N (parámetro) hilos en ejecución al mismo tiempo. Si hay más archivos que hilos permitidos, el programa deberá encolarlos hasta que puedan ser procesados. Se deberán tomar las precauciones necesarias (a criterio del grupo) para que el mismo archivo no sea procesado dos veces (a menos que sea puesto dos veces en el directorio).
La información que se espera obtener de cada archivo es, el nombre (en el momento en que se comienza a procesar) y la cantidad de cada uno de los caracteres graficables que lo componen (cuando se termina de procesar), por ejemplo:

```
./archivo1.txt
a: 200
b: 37
f: 44
[: 52
... etc
```

Se deberán omitir todos los caracteres que no tengan representación gráfica.
El programa deberá continuar en ejecución hasta que sea finalizado con las señales SIGINT o SIGTERM. Al finalizar, deberá informar el total de cada uno de los caracteres de todos los archivos procesados, ordenados de mayor a menor por cantidad de apariciones, más una lista de los archivos que quedaron en el directorio pero no llegaron a procesarse.

Tip: Para el monitoreo de directorios, investigue la herramienta inotify.

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
No excede la cantidad máxima de threads permitidos	Obligatorio
Procesa archivos que se agregan al directorio luego de haber iniciado el proceso	Obligatorio
Validación del tipo de espera usada cuando no hay más archivos por procesar	Obligatorio
Maneja correctamente la finalización normal y abrupta (kill)	Obligatorio
Código debidamente comentado	Deseable

Ejercicio 4:

Diseñe y codifique un programa que permita jugar al *memotest*. Debe tener una arquitectura cliente-servidor y capacidad para funcionar en diferentes computadoras, o sea, debe funcionar en una LAN.

Dinámica de juego
Un programa (servidor) funciona como tablero. Debe recibir las jugadas de parte del jugador (cliente) y responderle el resultado.
Otro programa (cliente) funciona como jugador. Este programa cliente permitirá al usuario ingresar las posiciones de la grilla a develar, indicando fila y columna, luego las envía al tablero (servidor) y recibe el resultado; determinando si tuvo éxito o fracaso. Después de cada jugada informa el resultado por pantalla.
La grilla es de 10 x 10 y en cada casillero se ocultan números naturales. Esta grilla es generada por el servidor, y cada cliente que se conecte utilizará la misma grilla.

Protocolo de comunicación
El jugador envía al tablero las coordenadas de las dos fichas que desea develar.
Ejemplo: f1=7, c1=2, f2=0, c2=9
El tablero responde con las coordenadas consultadas, junto al valor develado.
Ejemplo: f1=7, c1=2, v1=45, f2=0, c2=9, v2=45 (en este ejemplo, v1=v2 entonces éxito!)

Configuraciones
En el jugador: Debe recibir la IP y el puerto del servidor en cada jugada, por parámetros. -i y -p respectivamente.
En el tablero: Debe recibir el puerto por parámetro, -p.

Registro
Tanto el jugador como el tablero deben llevar un registro de las jugadas y de la comunicación en un archivo de registro, incluidas las direcciones IP, puertos involucrados y la hora en la que se produce.
El jugador debe registrar en un archivo el mensaje que le envía al tablero y la respuesta que recibe, y el resultado de la jugada.
El tablero debe registrar la distribución de las fichas al comienzo de su ejecución. También debe registrar los mensajes recibidos y los emitidos.

Criterios de corrección:

Control	Criticidad
Funciona correctamente según enunciado	Obligatorio
Makefile correcto y funcionando. Se compilará durante la corrección	Obligatorio
Maneja correctamente la concurrencia. El servidor podrá recibir múltiples conexiones	Obligatorio
Finaliza correctamente tanto el servidor como los clientes. No deja procesos zombies o huérfanos.	Obligatorio
Validación de los parámetros recibidos	Obligatorio
Maneja correctamente el cierre de los puertos al finalizar	Obligatorio
Permite pasar tanto IP como nombre de máquina	Deseable
Código debidamente comentado	Deseable

Ejercicio 5 (entrega tardía):

Diseñe e implemente un sistema que sea capaz de determinar si un número muy grande es primo. Deberá mostrar el resultado en pantalla.
Puesto que el cálculo de números primos requiere una gran cantidad de cálculos matemáticos, abordar el diseño desde un punto de vista monotarea podría resultar ineficiente. Por lo tanto, se pide que implemente una solución que deberá ser capaz de aprovechar el poder de cálculo de las computadoras vecinas (mínimo 2), para reducir el tiempo de cómputo. Aplicar la técnica de "divide y vencerás" puede serle de utilidad para resolverlo adecuadamente.

Considere que no está permitido contar con una lista de números primos.
Sugerimos discutir con los docentes el diseño del sistema, antes de comenzar con el desarrollo.

Deberá entregar un informe en donde compare varias ejecuciones (algunas distribuidas dentro del mismo equipo y otras distribuidas a varios equipos vecinos), con las diferencias encontradas y un análisis de lo hallado. Debe incluir una conclusión.

Criterios de corrección:

Control	Criticidad
Recibe el número a comprobar por parámetro	Obligatorio
Recibe detalles de configuración desde un archivo de texto plano (máquinas vecinas, etc)	Obligatorio
Recibe el nombre y ubicación del archivo de configuración por parámetro	Obligatorio
Se puede interrumpir el procesamiento manualmente	Obligatorio
Al finalizar la ejecución, no deja recursos bloqueados	Obligatorio
Justifica las decisiones de diseño e implementación con una breve descripción	Obligatorio
Infiere y adapta la mejor configuración (grado de paralelismo) para la ejecución, dependiendo del equipo donde se lo ejecuta.	Deseable
Muestra métricas de ejecución al finalizar	Deseable
Informe sobre comparativa de diferentes ejecuciones	Obligatorio

Trabajo Práctico Nro. 4 (GRUPAL):

- ***Formato de entrega: (***ver reglamento de entregas***)***
- ***Fecha de entrega:*** La fecha de presentación de los ejercicios está pactada en cada uno de los distintos grupos que componen el presente trabajo práctico. No se aceptará una entrega posterior sin tener cumplidas todas las fechas de entrega anteriores.
- ***Evaluación:*** La evaluación del trabajo práctico será realizada en conjunto entre el cuerpo docente asignado y el grupo desarrollador, por lo que será obligatoria la presencia de todos los integrantes del grupo en la fecha de entrega. En caso de encontrar errores se documentarán en la planilla de evaluación del TP y se les asignará una fecha en caso de ser necesaria una fecha de reentrega (en caso de no cumplir con dicha fecha de reentrega el práctico será desaprobado). Este trabajo práctico cuenta con coloquio obligatorio. Es importante indicar que dentro de la evaluación se tendrá en cuenta si se cumple o no con las peticiones de tener makefiles, la buena codificación y documentación del código fuente.

  Importante: El TP cuenta con una lista de validaciones mínimas, esto no implica que se puedan hacer otras validaciones al momento de evaluar el trabajo presentado
- ***Fecha de entrega:*** Del **17/11/2014 al 21/11/2014,** dependiendo del día que se curse la materia y sólo en la segunda fecha en que se cursa (miércoles / jueves / viernes) salvo para la cursada semi-presencial.
- ***Fecha tardía de entrega:*** Del **17/11/2014 al 21/11/2014, (NO ES ERROR, ES EL MISMO RANGO DE DÍAS)**.
- **Tipo de desarrollo*:*** El objetivo del presente trabajo práctico es el de programar un juego en formato gráfico para múltiples jugadores tanto en una como en varias máquinas al mismo tiempo. Es obligatorio para el desarrollo del TP el uso de procesos pesados, memoria compartida, Threads, SDL, y sockets para la implementación del juego. Además se debe poder ejecutar el cliente tanto en forma local como remota contra el mismo proceso servidor.

  No se podrán utilizar Threads de SDL ni semáforos System V
- **Objetivo:** Programar un juego basado en el juego "**Don King Kong"** denominado "**D-Don King Tournament"** para ser ejecutado en múltiples máquinas y multijugador. El objetivo de cada uno de los jugadores (**Mario**) será rescatar a la damisela en apuros (**Pauline**) sin que lo maten los tambores que arrojan el o los **Don King Kong** (la cantidad de monos queda a libre interpretación del grupo) o que lo toque el fuego que aparece aleatoriamente saliendo desde los barriles con fuego que se encuentran en la parte inferior de la pantalla.

  Para poder ganar la partida un Mario determinado deberá rescatar al menos 3 veces (en realidad la cantidad que se indique en un parámetro de inicialización del servidor – ver parámetros del servidor para más detalle -)

  Aleatoriamente en cualquier parte de la pantalla y sin importar si favorece o no a uno de los Mario's aparecerá un martillo que matará tanto a los fuegos como a los barriles que le son arrojados, pero tendrá como contra partida que durante el tiempo que Mario tenga el martillo no podrá subir escaleras.

  Es importante mencionar que salvo en el nivel inferior donde los Mario's no pueden cruzarse de la zona de juegos, en el resto de los lados si podrán

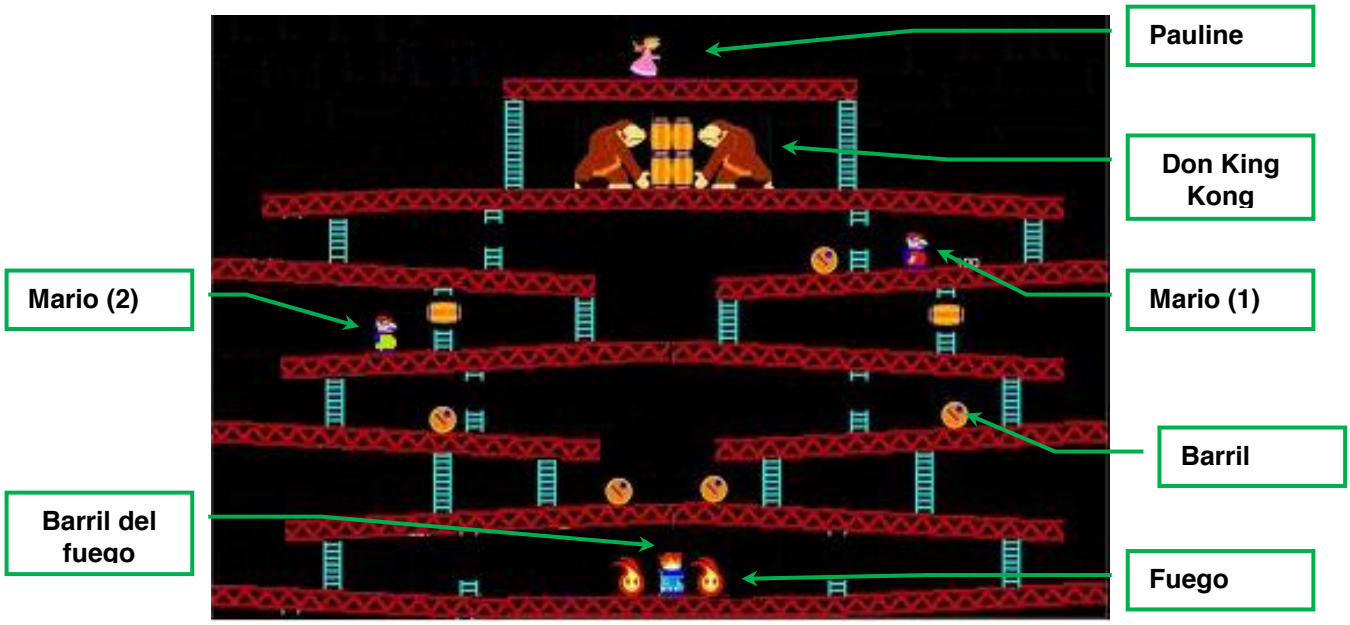
  El juego deberá tener tres niveles de juegos configurables, la propuesta es la siguiente:

  - **Nivel Brasil**: Se tiran pocos barriles y casi no salen fuegos.
  - **Nivel Normal:** La cantidad de barriles y de fuegos que hay en pantalla será la normal
  - **Nivel Extreme:** Caen barriles y fuegos indiscriminadamente.

  En la zona superior del juego (no visible en la pantalla de ejemplo) se deberá mostrar la información de:

  - Nombre de cada jugador (los **Marios**) que compiten
  - Cantidad de **Paulines** rescatadas por cada jugador
  - La cantidad de vidas perdidas hasta el momento

Imagen (sólo a nivel ilustrativo y podrá ser modificada por cada grupo):



Para mayor detalle se podrá ver un juego funcional para un jugador en:
http://www.juegosdonkey.com/donkeykongarcade.php

- **Arquitectura del juego:** el juego será un torneo **por simple eliminación** que se deberá jugar en múltiples máquinas y por múltiples jugadores. Al inicio del torneo el servidor se pondrá en modo escucha permitiendo que los clientes se conecten (infinitos clientes pudiendo ser par o impar la cantidad de conexiones). Una vez transcurrido el tiempo máximo de conexión el servidor de torneo no aceptará más conexiones y comenzará las partidas haciendo jugar a los jugadores según el orden de ingreso, así el primero que ingresó jugará contra el último que ingreso, el segundo que ingreso contra el anteúltimo y así sucesivamente hasta que no haya más jugadores y partidas para armar. En caso de que la cantidad de jugadores que se hayan anotado sean impares el último jugador que ingresó al torneo deberá ser eliminado, informando esta situación en el cliente.

  Los jugadores que vayan perdiendo deberán quedar conectados al servidor hasta que se declare a un ganador (lo que será informado en todas las pantallas que participaron de la partida salvo que el jugador haya abandonado).

  El servidor estará compuesto por un servidor principal (*Servidor del torneo*) que se encargará de llevar toda la administración del torneo, recibir las conexiones de los procesos clientes (Jugadores) y generar hijos (*Servidores de la partidas*) que se encargarán de administrar cada una de las partidas que se efectúan dentro del torneo. Tanto el servidor de partida como el servidor de torneo deberán ser binarios distintos y podrán o no estar emparentados al momento de su creación.

  Adicionalmente a los procesos anteriormente mencionados existirá un proceso cliente (*Jugador*) que se encargará de manejar la comunicación con el proceso servidor (en un principio con el servidor del torneo para inicializar la comunicación y luego con el servidor de partida que le fuera asignado), graficar el juego y aceptar los comandos del usuario que esté jugando la partida. Deberá existir un proceso cliente por cada jugador del torneo y podrán estar en la misma máquina que se ejecutó el servidor, o en máquinas distintas.

  Es importante mencionar que adicionalmente a los procesos descriptos anteriormente deberá existir un proceso monitor que asegure la correcta liberación de recursos en caso que por cualquier desperfecto (intencional o no) cualquiera de los procesos anteriormente detallados mueran. Este proceso monitor no sólo se ocupará de cerrar correctamente los recursos utilizados tanto en el servidor como en el torneo, sino que además deberá garantizarse que en caso de que el proceso que muera sea el propio monitor de alguna manera este será reejecutado ya sea por un proceso servidor o cliente.

  También se debe tener en cuenta que no se podrá contar con más de un proceso monitor por computadora, o sea que si en un mismo equipo se está ejecutando un proceso servidor y otro cliente o dos procesos clientes sólo deberá existir un proceso monitor para cada uno de los equipos computacionales que asegurará el correcto funcionamiento de los procesos que se ejecutan en esa máquina.

- ## Descripción detallada de los componentes:

Servidor del torneo: Será el proceso principal del juego. Sus funciones serán las siguientes:

- Recibir las conexiones de los nuevos clientes durante el período de inscripciones y luego de armar las partidas entre los jugadores siguiendo las modalidad explicada anteriormente.
- Asignar los clientes a los servidores de las partidas (ver detalles más adelante).
- Mostrar los resúmenes del torneo en la pantalla donde se ejecuta. Es importante mencionar que la información que se mostrará deberá estar en formato gráfico, informando al menos los siguientes puntos:
  - Detalle de las partidas jugadas y el ganador de cada una de ellas
  - Cantidad de clientes conectados
  - Partidas en curso
- Determinar y publicar las posiciones del torneo. La posición de cada jugador en el torneo deberá ser informada gráficamente en cada pantalla del jugador y la misma será la inversa de la ronda en la que fueron eliminados más uno (o sea el finalista será 2º, todos aquellos que perdieron en semifinales serán 3º y así sucesivamente)

Servidor de la partida: Serán hijos del servidor del torneo (proceso pesado en forma obligatoria), que se comunicará con el proceso padre a través de memoria compartida (obligatoriamente) y que tendrá como objetivo administrar una partida entre los dos jugadores que el servidor del torneo le asignó. Su funcionalidad deberá ser la siguiente:

- Comunicación con los clientes que participan de la partida (no se podrá genera un nuevo canal de comunicación, sino que deberá reutilizar los ya existentes)
- Administración del estado de la pantalla
- Llevar el resultado de la partida actual, compartirla con los clientes y actualizarla al servidor del torneo cada vez que hay un cambio (Ej. Mario rescato a Pauline)
- Este proceso deberá estar programado con threads que se encargarán de cada una de las funciones que tiene asignadas

Aclaración: Es importante mencionar que tanto el servidor de torneo como el de partida deberán ser binarios distintos.

Jugador: serán los procesos clientes y deberán estar programado con threads, teniendo que estar al menos divididas las siguientes funcionalidades:

- Comunicación con los servidores
- Dibujo de pantallas
- Lectura del teclado (por la funcionalidad del juego no se podrá utilizar el mouse)

Reglas del torneo:

- El servidor del torneo habilitará la inscripción durante un tiempo finito al inicio del mismo (parámetro del servidor de torneos)
- El torneo deberá ser obligatoriamente por eliminación directa respetando la modalidad de armado de las partidas indicada anteriormente
- Se declarará ganador de una partida a aquel jugador que haya rescatado la cantidad de veces que se indique en el archivo de configuración a la damisela en apuros.
- En caso de que el grupo así lo desee se podrá poner un límite de tiempo máximo para cada partida y en caso de que ninguno de los dos jugadores no logren rescatar a Pauline la cantidad de veces necesarias para ganar la partida el servidor de partida declarará a un ganador según un criterio seleccionado.
- Si durante la partida un jugador sale (ya sea intencionalmente o no), el jugador que quedó vivo será automáticamente el ganador de la partida, no siendo necesario continuar jugando.

Parámetros obligatorios del servidor:

En esta sección se detallan todos los parámetros que obligatoriamente deberá tener el servidor (del torneo / partida). Es obligatorio generar **un archivo de configuración** que contenga todos estos datos (no se tomará como válido pasar los valores como parámetros del binario).

- **Puerto del socket:** Indicará en qué puerto esperará el servidor por las conexiones desde los procesos cliente
- **Duración de la inscripción:** Será el tiempo que durará la inscripción al torneo expresado en segundos y deberá estar visible en la pantalla de control del servidor
- **Nivel de dificultad:** uno de los tres niveles explicados anteriormente
- **Cantidad de rescates:** Será la cantidad de veces que se deberá rescatar a la damisela en apuros para ser considerado ganador de la partida
- **Tiempo máximo:** (optativo) será el tiempo máximo que podrá tomar una partida antes que el servidor tome la decisión de declarar un ganador

Parámetros obligatorios de los clientes:

En esta sección se detallan todos los parámetros que obligatoriamente deberán tener los procesos clientes. Es obligatorio generar **un archivo de configuración** que contenga todos estos datos y que sea cargado dinámicamente no como parte de la compilación.

Comunicación y tiempo de espera:

- **IP / nombre del server:** Indicará la dirección IP o el nombre del servidor al que debe conectarse para iniciar el juego
- **Puerto del socket:** Indicará en qué puerto escucha el servidor
- **Teclas del juego:** Identificará qué teclas se deberán utilizar para jugar en cada uno de los clientes (se deberá poder jugar con distintas teclas para cada jugador)

Importante: No podrá ser ingresado como un parámetro el nombre del jugador, este dato deberá ser capturado por pantalla o enviado por parámetro al momento de ejecutar el cliente.

Lista de controles mínimos que se le aplicarán al desarrollo al momento de su entrega para evaluación:

- A continuación se detallan algunos de los controles que se le realizarán al juego al momento de ser sometido a su evaluación y cuál es la importancia de cada uno de ellos en la evaluación final.
- **Nota:** Esto es sólo una lista orientativa, pudiendo el grupo de docentes de la cátedra agregar nuevas condiciones sin previo aviso.

Tema	Control	Criticidad
Funcionalidad	Jugabilidad (es ameno para jugar y no existen Lag's)	Obligatorio
	Servidor muestra la información de las partidas en formato gráfico	Obligatorio
	Cumple con las reglas del juego	Obligatorio
	Declara ganador al jugador que rescato la cantidad de veces necesarias a Pauline	Obligatorio
	Mario muere cuando lo toca un barril o el fuego	Obligatorio
	Mario obtiene los martillos y mata los barriles y/o el fuego	Plus
	Tiene animación cuando Mario rescata a Pauline	Plus
	Muestra los nombres de los jugadores	Obligatorio
	El jugador puede salir en cualquier momento (no se considerará Ctrl-C como válido, debiendo implementar una tecla para salir)	Obligatorio
	Controla el tiempo para la inscripción al torneo	Obligatorio
	Se implementan los tres niveles con las dificultades mínimas planteadas	Obligatorio
	Tiene pantalla de presentación	Plus
	Muestra a las posiciones en cada uno de los clientes y en el servidor al momento de finalizar el torneo	Plus
Construcción	Funciona el cliente local	Obligatorio
	Funciona el cliente remoto	Obligatorio
	Funciona el servidor del torneo	Obligatorio
	Funciona el servidor de partidas	Obligatorio
	Funciona el monitor de recursos	Obligatorio
	Los servidores de torneo y partida son binarios distintos	Obligatorio
	Cantidad de Jugadores variable e ilimitada	Obligatorio
	Reacciona bien ante la finalización normal de un jugador	Obligatorio
	Utiliza Memoria Compartida entre servidor de partida y del torneo	Obligatorio
	Utiliza Threads en clientes y servidor de partida	Obligatorio
	Sincronización de los Threads	Obligatorio
	Cantidad de sockets limitada (comando netstat / ss)	Obligatorio
Parámetros	Teclas configurables	Obligatorio
	IP/Puerto	Obligatorio
	Tiempos y demás parámetros configurables	Obligatorio
	Funciona con nombres de máquinas	Plus
	Solicita el nombre del jugador en formato SDL	Plus
	Validar que al cambiar los parámetros funcione	Decrementa puntos si no se hace

Robustez	Reacción ante caídas de clientes o servidores	Obligatorio
	Liberación de recursos (Conexiones, Memoria, Semáforos)	Obligatorio
	No quedan procesos zombies ni huérfanos	Obligatorio
	Se puede matar con -9 a los dos servidores y monitor reacciona correctamente (nunca se evaluará matar al monitor y a los servidores o clientes al mismo tiempo)	Obligatorio
	Se puede matar normal o con -9 al monitor y el mismo es regenerado por el servidor y/o cliente	Obligatorio
	Consumo de cpu menor al 40% tanto en servidores como clientes	Obligatorio
	Consumo de cpu menor al 20% tanto en servidores como clientes	Plus
Codificación	Makefiles	Obligatorio
	No hay Implementación en los .h	Obligatorio
	Orden del código	Plus
	Código legible (Funciones pequeñas 10/15 líneas)	Plus
	No magic numbers. Se debe utilizar constantes para todo, los números y los strings.	Plus
	Separación funciones en .c/.o	Plus
	Programación orientada a objetos	Plus
	Comentarios y código autodocumentado	Plus