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
Equipo de Docentes: Boettner F., Catalano L.,
de Lizarralde R, Villamayor A
Auxiliares docentes: Loiacono F., Hirschfeldt D.,
Rodriguez A., Piubel F., Segura L.,

PLANIFICACIÓN Y GUÍA DE ACTIVIDADES PRÁCTICAS DE LABORATORIO (Primer cuatrimestre 2021)

Contenidos:

- Reglamento, Programa y Planificación
- Guía de Actividades Prácticas de Laboratorio

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 sí 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 2021

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.

Módulo 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

Módulo 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

Módulo 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) 2021

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: FUNDAMENTOS DE SISTEMAS OPERATIVOS
AUTOR: Gunnar Wolf, Esteban Ruiz, Federico Bergero y Erwin Meza
EDITORIAL: UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FECHA: 2015
OBSER.: Libro de referencia https://sistop.org/pdf/sistemas_operativos.pdf

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 Actividades Prácticas de Laboratorio para ser desarrollados por los alumnos en forma individual o 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.

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 actividades prácticas de laboratorio diseñadas especialmente para que el grupo de alumnos los programe.

Material Didáctico:

- Diapositivas Power Point de Clase sobre temas teóricos
- Guías de Actividades Prácticas de Laboratorio 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: Todas las Actividades Prácticas de Laboratorio que se realicen en ésta materia serán corregidos en un entorno GNU/Linux. Para mayor detalle de las versiones utilizadas concurrir al laboratorio 266.
En caso de que el alumnado decida realizar las actividades en otra plataforma, 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, la APL será considerado como no entregado.

Actividad Práctica de Laboratorio Nro. 1:

Scripting

Actividad Práctica de Laboratorio Nro. 2:

Scripting

Actividad Práctica de Laboratorio Nro. 3:

Programación de comunicación y procesos

Defensa y Recuperatorios

Todos los ejercicios 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

Tomando en cuenta la modalidad de cursada la metodología de evaluación será la siguiente:

Cursada presencial o semi-presencial: Se considerará que la cursada está bajo esta modalidad cuando durante todo el período de cursada se pueda al menos tener una reunión presencial en las instalaciones de la Universidad al mes durante todo el cuatrimestre.

* Esta asignatura 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 TPs, 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 las Trabajos Prácticos y las evaluaciones parciales serán el instrumento para medir el rendimiento y la aprobación de la cursada.

IMPORTANTE: Para la cursada correspondiente al primer cuatrimestre del 2021 se está analizando cambiar la modalidad de evaluación para la cursada virtual por lo que estaremos actualizando esta información en cuanto esté definida

Cursada virtual: Se considerará que la cursada está bajo esta modalidad cuando no se puede asistir a las instalaciones de la Universidad a realizar encuentros presenciales al menos una vez al mes durante el período de la cursada

* Esta asignatura 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 tres evaluaciones de aprendizaje (EA) durante el transcurso de la cursada virtual. Cada una de las EAs contarán con una instancia recuperatoria. La calificación de las EAs será **“Aprobado”**, **“Desaprobado”** o **“Ausente”** y permitirán al alumno a acceder a la siguiente instancia de validación denominada Evaluación de Validación (EdA)
* El contenido de las EAs será:
  * **EA 1:** todo el contenido teórico práctico y de laboratorio que se haya dictado hasta el momento de la evaluación
  * **EA 2:** todo el contenido teórico práctico y de laboratorio que se haya dictado desde la primera evaluación de aprendizaje y la segunda
  * **EA 3:** Será el contenido de las Actividades Prácticas de Laboratorio existentes en la presente guía. Se recomienda que la entrega de esta evaluación sea realizada tomando en cuenta las fechas indicadas en cada una de las partes.
* Además, se requiere una asistencia virtual registrada en Miel del 75% o más

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 o recuperatorios).

Cursada presencial o semi-presencial:

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 alcanza 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 y aprobar un examen final. La validez de la cursada será la vigente según las normas de la Universidad y el Departamento de Ingeniería.
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.**

Cursada virtual:

a) El alumno deberá tener aprobadas las tres EAs ya sea en su instancia inicial o su instancia de recuperación. Esto brindará la calificación de **cursada** le permitirá acceder a la instancia de EdA o al final
b) Aquel alumno que apruebe la Evaluación de Validación (EdA) quedará con la condición de **aprobada**, caso contrario mantendrá la calificación de **cursada** debiendo rendir un examen final para poder aprobar la materia
c) Aquel alumno que no haya aprobado las tres EA tendrá la condición de **ausente**, al igual que aquellos alumnos que no hayan alcanzado la meta del 75% de asistencia durante la cursada virtual

Régimen de Trabajos Prácticos (solo cursada presencial / semi-presencial / exámenes libres)

1- Para cumplimentar el régimen de Trabajos Prácticos el alumno deberá cumplimentar todos los contenidos presentados como Actividad Práctica de Laboratorio (APL) de la presente guía ya sea en forma grupal o individual
2- En fecha de entrega, se hará una corrección grupal de ejercicios en forma arbitraria para cada grupo.
3- La NO presentación o estar incompleto en la fecha propuesta significa su desaprobación en primera instancia para todos los integrantes del grupo).
4- Si bien la realización podrá ser grupal, su evaluación será individual a través de un examen escrito u oral.
5- El alumno que no apruebe la evaluación o NO haya entregado el contenido en fecha establecida, tendrá una nueva fecha para reentregar o rendir los contenidos no presentados / desaprobados (no aplica para exámenes libres)

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
- Cada alumno rendirá el final con el programa vigente al momento de rendir y no de su cursada

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 (normalmente la primera por disposición del departamento).

El contenido del examen final en modalidad Libre estará compuesto por:

a) **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 para rendir exámenes libres se deberán tener los trabajos prácticos ya vencidos aprobados:

Llamada	**TPs aprobados**
Julio 2021	Primer cuatrimestre 2021
Diciembre 2021	Segundo cuatrimestre 2021
Febrero / Marzo 2022	Segundo cuatrimestre 2021

Nota: En las fechas adicionales no se puede rendir en modalidad libre según reglamento académico.

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.

Para mayor detalle sobre la forma de rendir exámenes libres y los requerimientos a cumplir antes de presentarse comunicarse con el jefe de cátedras

CALENDARIO DE ACTIVIDADES

PLANIFICACIÓN DOCENTE PARA EL AÑO 2021

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 2021- (05/04/2021)

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 / EA1

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 / EA1

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 / EA1

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 / EA3

EVALUACIÓN PRÁCTICO 1: Presupuesto de tiempo: 1 Hs.

Solo para modalidad presencial o semi-presencial

Objetivos:

- Evaluar a los alumnos sobre los conocimientos adquiridos durante la actividad práctica 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 / EA1

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 / EA1

PRÁCTICO 2: Presupuesto de tiempo: 6 Hs.

Objetivos:

- Que el alumno incorpore los conocimientos en PowerSell necesarios

Tipo de conocimiento:

- Práctica de laboratorio

Evaluación del Módulo:

- Coloquio del TP / EA3

EVALUACIÓN PRÁCTICO 2: Presupuesto de tiempo: 1 Hs.

Solo para modalidad presencial o semi-presencial

Objetivos:

- Evaluar a los alumnos sobre los conocimientos adquiridos durante la actividad práctica 2

EVALUACIÓN 1: Presupuesto de tiempo: 4 Hs.

Solo para modalidad presencial o semi-presencial

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 / EA2

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 / EA2

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:

- Segundo parcial / EA2

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 / EA2

PRÁCTICO 3: Presupuesto de tiempo: 10 Hs.

Objetivos:

- Que el alumno incorpore los conocimientos necesarios para realizar procesos concurrentes en Linux

Tipo de conocimiento:

- Práctica de laboratorio

Evaluación del Módulo:

- Coloquio del TP / EA3

EVALUACIÓN PRÁCTICO 3: Presupuesto de tiempo: 1 Hs.

Solo para modalidad presencial o semi-presencial

Objetivos:

- Evaluar a los alumnos sobre los conocimientos adquiridos durante la actividad práctica 3

EVALUACIÓN 2: Presupuesto de tiempo: 4 Hs.

Solo para modalidad presencial o semi-presencial

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 / EA2

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 / EA2

RECUPERACIÓN 1: Presupuesto de tiempo: 4 Hs.

Solo para modalidad presencial o semi-presencial

Objetivos:

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

"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 primer cuatrimestre del ciclo lectivo 2021, 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
05/04/2021

Actividades Prácticas de Laboratorio

A continuación se detallan las Actividades Prácticas de Laboratorio (**APL**), su contenido, modalidad de presentación y condiciones de aprobación según el tipo de cursada:

Cada una de estas actividades prácticas formará parte del contenido formal requerido para aprobar o cursar la presente asignatura siendo estas parte conformante de:

- **Cursada presencial, semi-presencial o exámenes libres:** Trabajos prácticos
- **Cursada virtual:** Evaluación de Aprendizaje 3

Contenidos

APL 1:

Scripts con bash, awk, sed y señales en sistema operativo GNU/Linux

APL 2:

Scripts con PowerShell en Core de Linux

APL 3:

Desarrollo de procesos y comunicación en C / C++ bajo Linux

Modalidad de evaluación:

- La aprobación de cada APC se obtiene una vez que todos los ejercicios individuales estén aprobados de forma individual
- Para la aprobación individual de cada uno de los ejercicios se deberá al menos cumplir cada uno de los “criterios de corrección” obligatorios planteados por el cuerpo docente

Fechas de entrega:

- Cada APL tiene una **fecha de entrega de validación**, una **fecha de entrega final** y una **fecha de entrega para recuperación**
- Dependiendo de la modalidad de cursada estas fechas serán clasificadas en:
  - Obligatoria: El alumno deberá cumplir de forma obligatoria con esta fecha pactada, caso de no hacerlo tendrá la condición de desaprobado para todos los ejercicios contenidos en la APL
  - Recomendada: Se recomienda cumplir con esta fecha para poder continuar con el proceso evolutivo de evaluación de la APL
  - No Aplica: No aplica esa fecha para ese tipo de cursada
- Si una APL **no es entregada** en la **fecha de validación**, no se aceptará ninguna entrega total o parcial hasta su **entrega final** y/o su **entrega de recuperación** (dependiendo del tipo de cursada)
- Si una APL **fue entregada** en su **entrega de validación** podrá ser corregida y reentregada tantas veces como se requiera hasta su aprobación final hasta su fecha de **entrega de recuperación**

Reglamento de entrega:

Todas las APL tienen el mismo formato de entrega o reentregas que es descrito a continuación, y que en caso de no ser cumplido no será considerada como entregadas. Este formato de entrega contendrá al menos un paso obligatorio para cada modalidad de cursada:

Modalidad de cursada	Formato de entrega	Característica
Presencial o semi-presencial	Entrega carátula impresa	Obligatoria en Lab 266
	Entrega digital	Obligatoria
	Entrega presencial	Recomendada en Lab 266
Virtual	Entrega carátula formato digital	Obligatoria a través de MIEL en la sección EA3
	Entrega digital	Obligatoria en sisop y miel
	Entrega presencial	Recomendada reunión de teams
Exámenes libres	Entrega carátula formato digital	A través de correo electrónico al jefe de cátedras. Plazo máximo 15 días antes de la fecha de final en la que se desea rendir
	Entrega digital	

Entrega de carátula:

Por cada entrega o reentrega se deberá entregar una **carátula** utilizando para tal fin la plantilla disponible en el sitio web de la cátedra debidamente completada y firmada por cada uno de los integrantes del curso.

Las reentregas se realizarán con una nueva **carátula** conteniendo toda la información pertinente para que los docentes puedan realizar la corrección y deberá estar también firmada por todos los integrantes del grupo que sigan cursando la materia.

Importante: No se aceptarán **carátulas** no estén debidamente firmadas por todos los integrantes del grupo ya sea en modalidad impresa o virtual

Entrega digital:

Las APL deberán entregarse mediante la opción Entrega de APLs del sitio web de la cátedra (http://www.sisop.com.ar). Es importante tener en cuenta que la entrega la deberá realizar sólo uno de los integrantes de cada grupo. Queda a consideración de cada equipo de trabajo quién será el que realice las entregas.

El sistema aceptará un único archivo comprimido por cada entrega, con extensiones **zip, gzip, gz o tgz.**

El avance de las correcciones, feedback de los profesores y evaluaciones podrán visualizarlos todos los integrantes del grupo (sin importar quién hizo la entrega) desde la opción de menú Mis Notas.

Entrega presencial:

El cuerpo docente estará disponible para realizar la evaluación temprana de los trabajos durante la fecha de validación planificada para cada curso y modalidad de cursada.

El objetivo de esta evaluación temprana es que el grupo o al menos uno de los integrantes de cada grupo, realice la evaluación de cada uno de los ejercicios incluidos en la APL, y de esta manera el grupo tenga ya la información sobre si cada uno de los ejercicios está o no en condiciones de ser evaluado por el cuerpo docente.

La idea de esta evaluación temprana es que tanto el grupo docente como el equipo de trabajo se enteren de los problemas detectados en la solución brindada y el grupo pueda realizar las correcciones lo antes posible.

Modalidad de trabajo el día de entrega:

- Se recibirán carátulas para la evaluación de trabajos durante las dos primeras horas de la jornada
- Al menos un integrante de cada grupo debe estar presente para poder efectuar la corrección temprana

- El orden de recepción de las caratulas será utilizado como orden de atención de los grupos
- Un integrante del cuerpo docente llamará al grupo luego de haber descargado la APL desde el sitio web de entrega y en conjunto procederán a realizar las evaluaciones suministradas por el grupo de trabajo que se incluyan en la solución.
- El resultado de la evaluación realizada en cada uno de los ejercicios de la APL será el rechazo por no cumplir con los objetivos o la aceptación por parte del cuerpo docente lo que dará lugar a una prueba más exhaustiva a realizar solo por el cuerpo docente.
- En caso que un ejercicio sea desaprobado el equipo de trabajo deberá realizar la reentrega lo antes posible para que el cuerpo docente pueda realizar la evaluación.
- Las reentregas y APLs entregados en forma no presencial serán evaluados posteriormente a los entregados en forma presencial.

Nota: La aceptación del ejercicio no significa la aprobación ya que solo se realizarán las pruebas mínimas correspondientes a la entrega realizada por parte del equipo de trabajo. La aprobación final se brindará por el canal normal en los días posteriores a la entrega.

Actividad Práctica de Laboratorio Nro. 1:

- ***Tema:*** Programación de scripts en tecnología bash
- ***Descripción:*** Se programarán todos los scripts mencionados en el presente
- ***Formato de entrega:*** Siguiendo el protocolo especificado anteriormente
- ***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 número de APL al que pertenece y el número de ejercicio 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 de la APL 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.).
- ***Evaluación:*** Luego de entregado los docentes y ayudantes procederán a evaluar los ejercicios resueltos. Estas evaluaciones 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.

  Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el contenido presentado.

  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.

- ***Planificación:***

Tipo	Fecha Inicial	Fecha final	Cursada presencial y semi-presencial	Cursada Virtual	Exámenes Libres
Validación	28/04/2021	29/04/2021	Obligatoria	Recomendada	No Aplica
Entrega Final	05/07/2021	06/07/2021	No Aplica	Obligatoria	No Aplica
Recuperatorio	19/07/2021	20/07/2021	Obligatoria	Obligatoria	No Aplica

- ***Introducción:***

  El objetivo pedagógico del presente es que los alumnos adquieran un cierto entrenamiento sobre la programación de shell scripts, practicando el uso de utilitarios comunes provistos por los sistemas operativos (GNU/Linux).
  Todos los scripts, están orientados a la administración de una máquina, o red 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.
  Todos los scripts deberán funcionar en las instalaciones del laboratorio 266, o en instancias similares de instalación y versionado de bibliotecas, ya que es ahí donde serán controlados por el grupo docente.

- ***Ejercicios:***

  Tip: En caso de tener problemas con un script bajado desde un dispositivo 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:

Tomando en cuenta el siguiente script responda las preguntas que se encuentran más abajo.
Importante: como parte del resultado se deberá entregar el script en un archivo tipo sh y las respuestas en el mismo código.

```
#!/bin/bash
funcA() {
    echo "Error. La sintaxis del script es la siguiente:"
    echo "......................: $0 directorio 5" # COMPLETAR
}
funcB() {
    echo "Error. $1 ....................." # COMPLETAR
}
funcC() {
    if [[ ! -d $2 ]]; then
        funcB
    fi
}
funcC $# $1 $2 $3 $4 $5

LIST=$(ls -d $1*/)
ITEMS=()

for d in $LIST; do
    ITEM="`ls $d | wc -l`-$d"
    ITEMS+=($ITEM)
done

IFS=$'\n' sorted=($(sort -rV -t '-' -k 1 <<<${ITEMS[*]}))
CANDIDATES="${sorted[*]:0:$2}"
unset IFS
echo "....................." # COMPLETAR
printf "%s\n" "$(cut -d '-' -f 2 <<<${CANDIDATES[*]})"
```

Responda:

1. ¿Cuál es el objetivo de este script?, ¿Qué parámetros recibe?
2. Comentar el código según la funcionalidad (no describa los comandos, indique la lógica)
3. Completar los “echo” con el mensaje correspondiente.
4. ¿Qué nombre debería tener las funciones funcA, funcB, funcC?
5. ¿Agregaría alguna otra validación a los parámetros?, ¿existe algún error en el script?
6. ¿Qué información brinda la variable $#? ¿Qué otras variables similares conocen? Explíquelas.
7. Explique las diferencias entre los distintos tipos de comillas que se pueden utilizar en Shell scripts.
8. ¿Qué sucede si se ejecuta el script sin ningún parámetro?

Ejercicio 2:

Realice un script que dado un archivo de texto plano (pasado por parámetro) le aplique las siguientes reglas:

- Eliminar espacios duplicados.
- Eliminar espacios de más antes de un punto (.), coma (,) o punto y coma (;)
- Agregar un espacio luego de un punto seguido, coma o punto y coma (En caso de que no lo tuviera).

Generando un nuevo archivo a guardarse en el mismo directorio del archivo original, agregándole al final del nombre la fecha y hora en el formato:

[NOMBRE ORIGINAL]_[YYYYMMDDHHmm].[extensión, si la tuviera]

Además, debe generar por cada archivo un reporte de corrección, con el mismo nombre del archivo corregido, pero con extensión .log con los siguientes datos:

- Cantidad de correcciones realizadas.
- Cantidad de inconsistencias encontradas, diferenciadas por paréntesis dispares, signos de pregunta dispares y signos de admiración dispares. Llamamos dispares cuando se encuentra el signo o paréntesis de apertura y no el de cierre, o viceversa. Para verificar esto basta con tomar la diferencia entre los signos de apertura y cierre, sin mayor análisis sintáctico.

Ejemplo de llamada:

./Corrector.sh -in [archivo]

Nota: Se debe validar que el archivo pasado por parámetro sea de texto plano, sin importar la extensión que contenga.

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con -h, -? o -help explicando cómo se lo debe invocar	Obligatorio
Valida parámetros.	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Ejercicio 3:

En una determinada empresa se corre un proceso que genera archivos de texto plano de sus operaciones diarias, pero sucede que dicho proceso algunos días genera más de una copia del mismo archivo y hasta con diferentes nombres. Se requiere generar un script que pueda determinar que archivos son copia de otros archivos (por su contenido) para posteriormente tomar la decisión de cuales archivos se eliminaran. Además, se debe generar un archivo log en otro directorio donde se listen los archivos repetidos con su ubicación original, cuyo nombre será Resultado_[YYYYMMDDHHmm].out.
Dicho script debe recibir los siguientes parámetros:

- -Directorio: directorio donde se generan los archivos generados por el proceso. El mismo puede contener subdirectorios y los archivos encontrarse dentro o fuera de estos.
- -DirectorioSalida: directorio donde se guarda el archivo log, no debe ser el mismo directorio que -Directorio.
- -Umbral: tamaño en **KB** a partir del cual se empezarán a evaluar si los archivos presentan duplicados en el directorio ya que a veces el proceso a veces también genera archivos erróneos(vacíos) o incompletos y se desean filtrar del script.

Los mismos se deben poder recibir en cualquier orden.
Si por ejemplo tenemos 3 archivos “A”, “B”, “A” iguales y 2 archivos “D”, “E”. El script debería informar cada grupo de archivos duplicados delimitados por una línea vacía. Puede darse el caso que los archivos A B C se encuentran en subdirectorios diferentes y la salida del script debe ser la misma.
Ejemplo de salida (A, B, C son iguales entre sí. D, E son iguales entre sí):

```
A       /home/Entrada
B       /home/Entrada
A       /home/Entrada/SubDirectorio1
D       /Home/SubDirectorio2
E       /Home/SubDirectorio3
```

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con -h, -? o -help explicando cómo se lo debe invocar	Obligatorio
Valida parámetros (existencia de rutas y permisos, cantidad de parámetros).	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Ejercicio 4:

Un amigo suyo, muy ordenado y un poco obsesivo, tuvo la idea de organizar su carpeta de descargas por extensión. De esta manera, cada vez que descarga un nuevo archivo (es decir, un archivo es creado o copiado a la carpeta de descargas), lo busca y manualmente lo mueve a un directorio que tiene por nombre la extensión de dicho archivo. Por ejemplo, un archivo .zip irá a la carpeta “ZIP” y uno .pdf irá a “PDF”.

Como usted es una buena persona, decide inventar una solución para automatizar la tarea que realiza su amigo.

Se pide realizar un script demonio que detecte cada vez que un archivo nuevo aparezca en un directorio “descargas”. Una vez detectado, se debe mover a un subdirectorio “extensión” cuyo nombre será la extensión del archivo y que estará localizado en un directorio “destino”. Tener en cuenta que,

al ser un demonio, el script debe liberar la terminal una vez ejecutado, dejando al usuario la posibilidad de ejecutar nuevos comandos.

El script recibirá los siguientes parámetros:

- -d: Indica el directorio a monitorear (directorio “descargas”).
- -o: Indica el directorio que contendrá los subdirectorios “extensión” (directorio “destino”). Si no se pasa valor a este parámetro, el directorio “destino” será el de “descarga”.
- -s: Si está presente, se debe detener el demonio. No puede pasarse al mismo tiempo que los otros parámetros.

Para tener en cuenta:

- Pueden existir archivos sin extensión. En ese caso, deberán moverse al directorio “destino” (fuera de los subdirectorios “extensión”).
- Los archivos con nombres que empiecen por “.” y que no tengan otra extensión deben trataste como los archivos sin extensión. Ejemplos: “.gitignore”, “.mailconfig”.
- Las extensiones no necesariamente tienen que ser de 3 caracteres. Ejemplos: .c, .cs, .cpp, .csproj.
- Si el subdirectorio “extensión” no existe, deberá crearse.
- Los subdirectorios “extensión” deben tener el nombre en mayúscula.

Criterios de corrección:

Control	**Criticidad**
El script ofrece ayuda con -h, -? o -help explicando cómo se lo debe invocar	Obligatorio
Valida parámetros (existencia de rutas y permisos, cantidad de parámetros).	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Ejercicio 5:

Cada vez que llegaba una fecha de final, una universidad tenía el mismo problema: todas las materias dictadas entregaban las actas de final en papel para luego ser cargadas manualmente en el sistema central de actas. Este proceso, además de lento, contaba con un elevado número de errores, en su mayoría involuntarios.

Para agilizar este proceso, se decidió crear un sistema al que los profesores pudieran mandar las actas de manera digital. El flujo del sistema es el siguiente:

- Luego de tomar un final, el jefe de cátedra de cada materia crea un archivo en formato CSV con una fila por alumno y en donde se informa DNI y las notas de cada uno de los puntos del examen (B, R o M).
- Este archivo se sube a una web y es almacenado en un directorio a la espera de su procesamiento.
- Cada sábado a las 23:45hs, corre un script que procesa todos los CSV subidos (uno por mesa de examen) y genera un archivo JSON con las notas de los alumnos que rindieron final esa semana.
- El archivo JSON es luego leído por el sistema central de actas.

Como grupo de alumnos que participa de un voluntariado de ayuda en la universidad, se ofrecen para programar un script que realice el procesamiento de las notas y genere el archivo JSON.

El script recibirá los siguientes parámetros:

- --notas: Directorio en el que se encuentran los archivos CSV.
- --salida: Ruta del archivo JSON a generar (incluye nombre del archivo).

Para tener en cuenta:

- Cada archivo CSV representa una fecha de final, por lo tanto, tendrá las notas de todos los ejercicios de los alumnos que se presentaron a rendir. Los ausentes no están listados.
- El nombre del archivo CSV es el código de la materia y la fecha de final, separados por un “_”. Ej: “1115_20210317.csv”.
- La cantidad de ejercicios por final es variable entre materias (una materia siempre tiene la misma cantidad de ejercicios). Esto quiere decir que la cantidad de columnas que tiene cada archivo CSV no tiene porqué ser la misma.
- Se genera un único archivo JSON que contiene la información de todas las materias y alumnos que rindieron durante la semana. Es decir, puede haber materias y alumnos repetidos. Sin embargo, un alumno no puede rendir la misma materia más de una vez a la semana, pero una materia puede tomar más de un final en la semana.

Para calcular la nota de cada final, se debe tener en cuenta que:

- Cada ejercicio tiene el mismo peso en la nota final. Para calcularlo se puede usar la siguiente fórmula: 10 / CantidadEjercicios.
- Un ejercicio bien (B) vale el ejercicio entero.
- Un ejercicio regular (R) vale medio ejercicio.
- Un ejercicio mal (M) no suma puntos a la nota final.

Formato de los archivos

CSV (no se debe incluir la primera fila de encabezados, se muestra solo para explicar el ejemplo):

```
Dni,Ej-1,Ej-2,Ej-3,…,Ej-n
12345678,b,m,r,…,m
87654321,b,b,b,…,r
```

JSON:

```
{ "actas": [
  {
    "dni": "12345678",
    "notas": [
      { "materia": 1115, "nota": 8 },
      { "materia": 1116, "nota": 2 }
    ]
  },
  {
    "dni": "87654321",
    "notas": [
      { "materia": 1116, "nota": 9 },
      { "materia": 1118, "nota": 7 }
    ]
  }
] }
```

El archivo JSON generado debe ser un JSON válido. No interesa el formato (tabs, espacios, saltos de línea) siempre y cuando sea válido. Para validarlo se pueden usar sitios web como https://jsonformatter.curiousconcept.com/.

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con -h, -? o -help explicando cómo se lo debe invocar	Obligatorio
Valida parámetros (existencia de rutas y permisos, cantidad de parámetros).	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se debe utilizar AWK para la lectura de los archivos CSV	Obligatorio
Se implementan funciones	Deseable

Ejercicio 6:

Realizar un script que emule el comportamiento del comando rm, pero utilizando el concepto de "papelera de reciclaje", es decir que, al borrar un archivo se tenga la posibilidad de recuperarlo en el futuro.

El script tendrá las siguientes opciones:

- **-l** listar los archivos que contiene la papelera de reciclaje, informando nombre de archivo y su ubicación original.
- **-r [archivo]** recuperar el archivo pasado por parámetro a su ubicación original.
- **-e** vaciar la papelera de reciclaje (eliminar definitivamente)
- **[archivo]** Sin modificador para que elimine el archivo (o sea, que lo envíe a la papelera de reciclaje).

La papelera de reciclaje deberá ser un archivo comprimido ZIP y debe estar alojada en el home del usuario que ejecuta el comando, en caso de no encontrarse debe crearla.

Nota1: Tenga presente que archivos de diferentes directorios podrían tener el mismo nombre. El script debe considerar estos casos.

Nota2: En caso de que se quiera recuperar un nombre de archivo que esta varias veces en la papelera, debe listar los archivos y su ubicación original y preguntar cuál se quiere recuperar.
Ejemplo:

```
$ ./papelera.sh -l

Pepe        /home/usuario1/docs
Paco        /home/usuario1/docs
pepe        /home/usuario1/temp
Pepe        /home/usuario1/descargas
Animales    /home/usuario1/imágenes
Pepe        /home/usuario1/imágenes
…

$ ./papelera.sh -r Pepe

1 - Pepe    /home/usuario1/docs
2 - Pepe    /home/usuario1/descargas
3 - Pepe    /home/usuario1/imágenes
¿Qué archivo desea recuperar? ___

1
```

(Recupera el archivo Pepe a /home/usuario1/docs)

Criterios de corrección:

Control	**Criticidad**
El script ofrece ayuda con -h, -? o -help explicando cómo se lo debe invocar	Obligatorio
Valida parámetros.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Actividad Práctica de Laboratorio Nro. 2:

- ***Tema:*** Programación de scripts básicos en PowerShell
- ***Descripción:*** Se programarán todos los scripts mencionados en el presente
- ***Formato de entrega:*** Según el protocolo especificado anteriormente
- ***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 número de APL al que pertenece y el número de ejercicio 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 de la APL 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.).
- ***Evaluación:*** Luego de entregado los docentes y ayudantes procederán a evaluar los ejercicios resueltos. Estas evaluaciones 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.

  Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el contenido presentado.

  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.

- ***Planificación:***

Tipo	Fecha Inicial	Fecha final	Cursada presencial y semi-presencial	Cursada Virtual	Exámenes Libres
Validación	19/05/2021	20/05/2021	Obligatoria	Recomendada	No Aplica
Entrega Final	05/07/2021	06/07/2021	No Aplica	Obligatoria	No Aplica
Recuperatorio	19/07/2021	20/07/2021	Obligatoria	Obligatoria	No Aplica

- ***Introducción:***

  El objetivo pedagógico del presente es que los alumnos adquieran un cierto entrenamiento sobre la programación de shell scripts en lenguaje PowerSell.
  Todos los scripts están orientados a la administración de una máquina, o red 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 **administrator**, 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, o en instancias similares de instalación y versionado de bibliotecas, ya que es ahí donde serán controlados por el grupo docente.

- ***Ejercicios:***

  Importante: la versión mínima a utilizar para confeccionar y evaluar este TP debe ser Powershell 6, ya que es la versión de Powershell Core donde se realizará la corrección

Ejercicio 1:

Tomando en cuenta el siguiente script responda las preguntas que se encuentran más abajo.
Importante: como parte del resultado se deberá entregar el script en un archivo tipo sh y las respuestas en el mismo código.

```
[CmdletBinding()]
Param (
    [Parameter(Position = 1, Mandatory = $false)]
    [ValidateScript( { Test-Path -PathType Container $_ } )]
    [String] $param1,
    [int] $param2 = 0
)

$LIST = Get-ChildItem -Path $param1 -Directory
$ITEMS = ForEach ($ITEM in $LIST) {
    $COUNT = (Get-ChildItem -Path $ITEM).Length
    $props = @{
        name  = $ITEM
        count = $COUNT
    }
    New-Object psobject -Property $props
}

$CANDIDATES = $ITEMS | Sort-Object -Property count -
Descending | Select-Object -First $param2 | Select-Object -
Property name

Write-Output "....................." # COMPLETAR
$CANDIDATES | Format-Table -HideTableHeaders
```

Responda:

1. ¿Cuál es el objetivo de este script?, ¿Qué parámetros recibe?, renombre los parámetros con un nombre adecuado.
2. Comentar el código según la funcionalidad (no describa los comandos, indique la lógica)
3. Completar el Write-Output con el mensaje correspondiente.
4. ¿Agregaría alguna otra validación a los parámetros?, ¿existe algún error en el script?
5. ¿Para qué se utiliza [CmdletBinding()]?
6. Explique las diferencias entre los distintos tipos de comillas que se pueden utilizar en Shell scripts.
7. ¿Qué sucede si se ejecuta el script sin ningún parámetro?

Ejercicio 2:

Realice un script que dado un archivo de texto plano (pasado por parámetro) le aplique las siguientes reglas:

- Eliminar espacios duplicados.
- Eliminar espacios de más antes de un punto (.), coma (,) o punto y coma (;)
- Agregar un espacio luego de un punto seguido, coma o punto y coma (En caso de que no lo tuviera).

Generando un nuevo archivo a guardarse en el mismo directorio del archivo original, agregándole al final del nombre la fecha y hora en el formato:

[NOMBRE ORIGINAL]_[YYYYMMDDHHmm].[extensión, si la tuviera]

Además, debe generar por cada archivo un reporte de corrección, con el mismo nombre del archivo corregido, pero con extensión .log con los siguientes datos:

- Cantidad de correcciones realizadas.
- Cantidad de inconsistencias encontradas, diferenciadas por paréntesis dispares, signos de pregunta dispares y signos de admiración dispares. Llamamos dispares cuando se encuentra el signo o paréntesis de apertura y no el de cierre, o viceversa. Para verificar esto basta con tomar la diferencia entre los signos de apertura y cierre, sin mayor análisis sintáctico.

Ejemplo de llamada:

./Corrector.ps1 -in [archivo]

Nota: Se debe validar que el archivo pasado por parámetro sea de texto plano, sin importar la extensión que contenga.

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con Get-Help.	Obligatorio
Valida parámetros.	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Ejercicio 3:

En una determinada empresa se corre un proceso que genera archivos de texto plano de sus operaciones diarias, pero sucede que dicho proceso algunos días genera más de una copia del mismo archivo y hasta con diferentes nombres. Se requiere generar un script que pueda determinar que archivos son copia de otros archivos (por su contenido) para posteriormente tomar la decisión de cuales archivos se eliminaran. Además, se debe generar un archivo log en otro directorio donde se listen los archivos repetidos con su ubicación original, cuyo nombre será Resultado_[YYYYMMDDHHmm].out.
Dicho script debe recibir los siguientes parámetros:

- -Directorio: directorio donde se generan los archivos generados por el proceso. El mismo puede contener subdirectorios y los archivos encontrarse dentro o fuera de estos.
- -DirectorioSalida: directorio donde se guarda el archivo log, **no** debe ser el mismo directorio que -Directorio.
- -Umbral: tamaño en **KB** a partir del cual se empezarán a evaluar si los archivos presentan duplicados en el directorio ya que a veces el proceso también genera archivos erróneos (vacíos) o incompletos y se desean filtrar del script.

Los mismos se deben poder recibir en cualquier orden.
Si por ejemplo tenemos 3 archivos “A”, “B”, “A” iguales y 2 archivos “D”, “E”. El script debería informar cada grupo de archivos duplicados delimitados por una línea vacía. Puede darse el caso que los archivos A B C se encuentran en subdirectorios diferentes y la salida del script debe ser la misma.
Ejemplo de salida (A, B, C son iguales entre sí. D, E son iguales entre sí):

```
A       /home/Entrada
B       /home/Entrada
A       /home/Entrada/SubDirectorio1
D       /Home/SubDirectorio2
E       /Home/SubDirectorio3
```

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con Get-Help.	Obligatorio
Valida parámetros.	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Ejercicio 4:

Un amigo suyo, muy ordenado y un poco obsesivo, tuvo la idea de organizar su carpeta de descargas por extensión. De esta manera, cada vez que descarga un nuevo archivo (es decir, un archivo es creado o copiado a la carpeta de descargas), lo busca y manualmente lo mueve a un directorio que tiene por nombre la extensión de dicho archivo. Por ejemplo, un archivo .zip irá a la carpeta “ZIP” y uno .pdf irá a “PDF”.

Como usted es una buena persona, decide inventar una solución para automatizar la tarea que realiza su amigo.

Se pide realizar un script que haciendo uso de eventos detecte cada vez que un archivo nuevo aparezca en un directorio “descargas”. Una vez detectado, se debe mover a un subdirectorio “extensión” cuyo nombre será la extensión del archivo y que estará localizado en un directorio “destino”. Como requisito adicional, el script debe liberar la terminal una vez ejecutado, dejando al usuario la posibilidad de correr nuevos comandos.

El script recibirá los siguientes parámetros:

- -Descargas: Indica el directorio a monitorear (directorio “descargas”).
- -Destino: Indica el directorio que contendrá los subdirectorios “extensión” (directorio “destino”). Si no se pasa valor a este parámetro, el directorio “destino” será el de “descarga”.
- -Detener: Parámetro tipo switch. Si está presente, se debe detener la detección de archivos. No puede pasarse al mismo tiempo que los otros parámetros.

Para tener en cuenta:

- Pueden existir archivos sin extensión. En ese caso, deberán moverse al directorio “destino” (fuera de los subdirectorios “extensión”).
- Los archivos con nombres que empiecen por “.” y que no tengan otra extensión deben trataste como los archivos sin extensión. Ejemplos: “.gitignore”, “.mailconfig”.
- Las extensiones no necesariamente tienen que ser de 3 caracteres. Ejemplos: .c, .cs, .cpp, .csproj.
- Si el subdirectorio “extensión” no existe, deberá crearse.
- Los subdirectorios “extensión” deben tener el nombre en mayúscula.

Criterios de corrección:

Control	**Criticidad**
El script ofrece ayuda con Get-Help.	Obligatorio
Valida parámetros.	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se deben utilizar los eventos de FileSystemWatcher para la resolución del ejercicio	Obligatorio
Se implementan funciones	Deseable

Ejercicio 5:

Cada vez que llegaba una fecha de final, una universidad tenía el mismo problema: todas las materias dictadas entregaban las actas de final en papel para luego ser cargadas manualmente en el sistema central de actas. Este proceso, además de lento, contaba con un elevado número de errores, en su mayoría involuntarios.

Para agilizar este proceso, se decidió crear un sistema al que los profesores pudieran mandar las actas de manera digital. El flujo del sistema es el siguiente:

- Luego de tomar un final, el jefe de cátedra de cada materia crea un archivo en formato CSV con una fila por alumno y en donde se informa DNI y las notas de cada uno de los puntos del examen (B, R o M).
- Este archivo se sube a una web y es almacenado en un directorio a la espera de su procesamiento.
- Cada sábado a las 23:45hs, corre un script que procesa todos los CSV subidos (uno por mesa de examen) y genera un archivo JSON con las notas de los alumnos que rindieron final esa semana.
- El archivo JSON es luego leído por el sistema central de actas.

Como grupo de alumnos que participa de un voluntariado de ayuda en la universidad, se ofrecen para programar un script que realice el procesamiento de las notas y genere el archivo JSON.

El script recibirá los siguientes parámetros:

- -Notas: Directorio en el que se encuentran los archivos CSV.
- -Salida: Ruta del archivo JSON a generar (incluye nombre del archivo).

Para tener en cuenta:

- Cada archivo CSV representa una fecha de final, por lo tanto, tendrá las notas de todos los ejercicios de los alumnos que se presentaron a rendir. Los ausentes no están listados.
- El nombre del archivo CSV representa el código de la materia y la fecha. Ej: “1115_20210317.csv”.
- La cantidad de ejercicios por final es variable entre materias (una materia siempre tiene la misma cantidad de ejercicios). Esto quiere decir que la cantidad de columnas que tiene cada archivo CSV no tiene porqué ser la misma.
- Se genera un único archivo JSON que contiene la información de todas las materias y alumnos que rindieron durante la semana. Es decir, puede haber materias y alumnos repetidos. Sin embargo, un alumno no puede rendir la misma materia más de una vez a la semana, pero una materia puede tomar más de un final en la semana.

Para calcular la nota de cada final, se debe tener en cuenta que:

- Cada ejercicio tiene el mismo peso en la nota final. Para calcularlo se puede usar la siguiente fórmula: 10 / CantidadEjercicios.
- Un ejercicio bien (B) vale el ejercicio entero.
- Un ejercicio regular (R) vale medio ejercicio.
- Un ejercicio mal (M) no suma puntos a la nota final.

Formato de los archivos

CSV (no se debe incluir la primera fila de encabezados, se muestra solo para explicar el ejemplo):

```
Dni,Ej-1,Ej-2,Ej-3,…,Ej-n
12345678,b,m,r,…,m
87654321,b,b,b,…,r
```

JSON:

```
{ "actas": [
  {
    "dni": "12345678",
    "notas": [
      { "materia": 1115, "nota": 8 },
      { "materia": 1116, "nota": 2 }
    ]
  },
  {
    "dni": "87654321",
    "notas": [
      { "materia": 1116, "nota": 9 },
      { "materia": 1118, "nota": 7 }
    ]
  }
] }
```

El archivo JSON generado debe ser un JSON válido. No interesa el formato (tabs, espacios, saltos de línea) siempre y cuando sea válido. Para validarlo se pueden usar sitios web como https://jsonformatter.curiousconcept.com/.

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con Get-Help.	Obligatorio
Valida parámetros.	Obligatorio
Se adjuntan archivos de prueba por parte del grupo.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se deben utilizar los cmdlets Import-Csv y ConvertTo-Json de Powershell	Obligatorio
Se implementan funciones	Deseable

Ejercicio 6:

Realizar un script que emule el comportamiento del comando rm, pero utilizando el concepto de "papelera de reciclaje", es decir que, al borrar un archivo se tenga la posibilidad de recuperarlo en el futuro.
El script tendrá las siguientes opciones:

- **-l** listar los archivos que contiene la papelera de reciclaje, informando nombre de archivo y su ubicación original.
- **-r [archivo]** recuperar el archivo pasado por parámetro a su ubicación original.
- **-e** vaciar la papelera de reciclaje (eliminar definitivamente)
- **[archivo]** Sin modificador para que elimine el archivo (o sea, que lo envíe a la papelera de reciclaje).

La papelera de reciclaje deberá ser un archivo comprimido ZIP y debe estar alojada en el home del usuario que ejecuta el comando, en caso de no encontrarse debe crearla.

Nota1: Tenga presente que archivos de diferentes directorios podrían tener el mismo nombre. El script debe considerar estos casos.
Nota2: En caso de que se quiera recuperar un nombre de archivo que esta varias veces en la papelera, debe listar los archivos y su ubicación original y preguntar cuál se quiere recuperar.
Ejemplo:

```
$ ./papelera.ps1 -l

Pepe          /home/usuario1/docs
Paco          /home/usuario1/docs
pepe          /home/usuario1/temp
Pepe          /home/usuario1/descargas
Animales      /home/usuario1/imágenes
Pepe          /home/usuario1/imágenes
…

$ ./papelera.ps1 -r Pepe

1 - Pepe      /home/usuario1/docs
2 - Pepe      /home/usuario1/descargas
3 - Pepe      /home/usuario1/imágenes
¿Qué archivo desea recuperar? ___

1
```

(Recupera el archivo Pepe a /home/usuario1/docs)

Criterios de corrección:

Control	Criticidad
El script ofrece ayuda con Get-Help.	Obligatorio
Valida parámetros.	Obligatorio
Funciona correctamente según enunciado.	Obligatorio
Funciona con rutas relativas, absolutas o con espacios.	Obligatorio
Se implementan funciones	Deseable

Actividad Práctica de Laboratorio Nro. 3:

- ***Tema:*** Procesos, comunicación y sincronización
- ***Descripción:*** Codificar todos los programas mencionados en el presente trabajo en C o C++ bajo plataforma GNU/Linux
- ***Formato de entrega:*** Según el protocolo especificado anteriormente
- ***Documentación:*** Todos los programas que se entreguen deben tener un encabezado y un fin de archivo. Dentro del encabezado deben figurar el nombre del programas, el número de APL al que pertenece y el número de ejercicio 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 de la APL 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.).
- ***Evaluación:*** Luego de entregado los docentes y ayudantes procederán a evaluar los ejercicios resueltos. Estas evaluaciones 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.

  Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el contenido presentado.

  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.

- ***Planificación:***

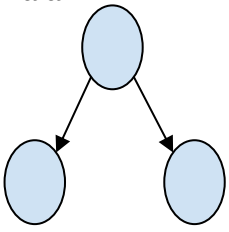
Tipo	Fecha Inicial	Fecha final	Cursada presencial y semi-presencial	Cursada Virtual	Exámenes Libres
Validación	23/06/2021	24/06/2021	Obligatoria	Recomendada	No Aplica
Entrega Final	05/07/2021	06/07/2021	No Aplica	Obligatoria	No Aplica
Recuperatorio	19/07/2021	20/07/2021	Obligatoria	Obligatoria	No Aplica

- ***Ejercicios:***

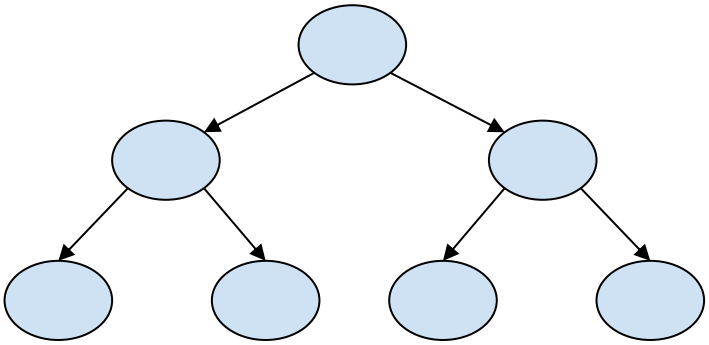
  Ejercicio 1:

  Realizar un programa que genere un árbol de procesos de N generaciones (siendo N un valor entero recibido por parámetro). Cada proceso deberá generar dos procesos hijos por cada generación. Algunos ejemplos de grafos son:

  Para N=2



Para N=3



Además cada proceso deberá indicar todo el árbol genealógico superior. En el ejemplo del último caso será:

Proceso 2: Pid 1
Proceso 3: Pid 1
Proceso 4: Pid 2, Pid 1
Proceso 5: Pid 2, Pid 1
Proceso 6: Pid 3, Pid 1
Proceso 7: Pid 3, Pid 1

Poner una espera en los procesos (por ejemplo el clásico “presione cualquier tecla para continuar...”) antes de su finalización para permitir la visualización del árbol desde otra terminal.

Criterios de corrección:

Control	**Criticidad**
Compila sin errores con el makefile entregado	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Existe algún tipo de ayuda para la ejecución de los procesos (-h, --help)	Obligatorio
Finalizan correctamente todos los procesos	Obligatorio
Valida que la variable N sea un número entero mayor a 1	Obligatorio

Ejercicio 2:

Desarrollar un programa que reciba por parámetro un número N (cantidad de iteraciones) y un número P (nivel de paralelismo). Se desea ejecutar un ciclo de N iteraciones matemáticas para cada número del 2 al 9. El ciclo está compuesto por las siguientes operaciones.

Suponiendo a la variable M como un número del intervalo mencionado (del 2 a 9) se debe:

1. Iniciar una variable acumulador con el valor M inicial
2. En la primera iteración se multiplica el contenido del acumulador por el valor de M
3. En la segunda iteración se suma el contenido del acumulador por sí mismo
4. En la tercera iteración se divide el contenido del acumulador por el número N
5. Repite desde el paso 2 hasta cumplir con los N ciclos.

Ejemplo con el valor M=2:

Paso 1: acumulador=2 //contenido de acumulador=2
Paso 2: acumulador*=2 //contenido de acumulador=4
Paso 3: acumulador+=acumulador //contenido de acumulador=8
Paso 4: acumulador/=2 //contenido de acumulador=4
Paso 5: acumulador*=2 //contenido de acumulador=8
etc….

Luego con M=3, M=4… hasta M=9. El nivel de paralelismo indicará la cantidad de threads que estarán operando en cada ciclo. Por ejemplo:

Nivel paralelismo P=1
Thread 1: opera sobre el número 2
Thread 1: opera sobre el número 3
Thread 1: opera sobre el número 4
Thread 1: opera sobre el número 5
Thread 1: opera sobre el número 6
Thread 1: opera sobre el número 7
Thread 1: opera sobre el número 8
Thread 1: opera sobre el número 9

Nivel paralelismo P=2

Thread	1:	opera	sobre	el	número	2
Thread	2:	opera	sobre	el	número	3
Thread	1:	opera	sobre	el	número	4
Thread	2:	opera	sobre	el	número	5
Thread	1:	opera	sobre	el	número	6
Thread	2:	opera	sobre	el	número	7
Thread	1:	opera	sobre	el	número	8

Thread 2: opera sobre el número 9

Al finalizar se deberá exponer por pantalla la cantidad de tiempo demorado en ejecutar cada ciclo con cada número y el tiempo final transcurrido desde el inicio de la ejecución hasta su finalización.

Criterios de corrección:

Control	**Criticidad**
Compila sin errores con el makefile entregado	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Existe algún tipo de ayuda para la ejecución de los procesos (-h, --help)	Obligatorio
Utiliza la biblioteca pthreads	Obligatorio
Valida correctamente los datos de entrada	Obligatorio

Ejercicio 3:

Se posee una estructura de archivos donde cada directorio lleva como nombre el año de facturación (2020, 2021, etc.) y dentro de los mismos se encuentra un archivo por cada mes facturado (enero.txt, febrero.txt,etc):
Cada archivo posee una línea con el monto de cada factura realizada, en todos los casos cada valor contendrá 2 valores para la parte decimal:
ejemplo: archivo 2020/enero.txt, contenido:
3244.23
444.22
556.00

Se deberán generar dos procesos no emparentados que se comunicarán a través de una tubería. El proceso A recibirá por parámetro la ruta donde se encuentran los directorios de facturación, mientras que el proceso B le mostrará al usuario un menú con las siguiente opciones:

1. Facturación mensual: Le solicitará el ingreso de año y mes de facturación para obtener el total facturado
2. Facturación anual: Le solicitará el ingreso de un año y obtendrá el total facturado en el mismo
3. Facturación media anual: Le solicitará el ingreso de un año y obtendrá la facturación media del año (operación matemática = totalFacturado/mesesFacturadosEnElAño).
4. Salir

Una vez que el usuario ingresa los datos, el proceso B le indicará la acción a realizar juntos con los parámetros al proceso A mediante una tubería. El proceso A llevará adelante la tarea pertinente y le devolverá al proceso B (también mediante tubería) el resultado a mostrar por pantalla.

Criterios de corrección:

Control	**Criticidad**
Compila sin errores con el makefile entregado	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Existe algún tipo de ayuda para la ejecución de los procesos (-h, --help)	Obligatorio
Cierra todos los procesos correctamente	Obligatorio
Valida correctamente los datos de entrada	Obligatorio
Adjunta un juego de archivos de prueba con la entrega	Obligatorio

Ejercicio 4:

Implementar el juego del ahorcado (Hangman) para la consola de Linux. Para ello deberá crear dos procesos no emparentados que se comuniquen a través de memoria compartida y sincronicen con semáforos POSIX.
Debe existir un proceso “Cliente”, cuya tarea será mostrar por pantalla el estado actual del juego y leer desde teclado las letras que el usuario vaya ingresando. También existirá un proceso “Servidor”, que será el encargado de la lógica, es decir; actualizar el estado del juego en base a las letras ingresadas, controlar si se gana o pierde la partida, etc.

A continuación se detallan algunas cuestiones para el diseño:
Servidor:

- Se ejecutará y quedará a la espera de que un cliente se ejecute.
- Solo podrá existir una instancia.
- Finalizará su ejecución al recibir la señal SIGUSR1, siempre y cuando no haya ninguna partida en curso.
- Deberá configurar una nueva partida al recibir la conexión de un nuevo cliente.
- La palabra a adivinar será seleccionada de manera aleatoria, elegida de un conjunto de palabras posibles almacenadas en un archivo de texto plano.
- Deberá ignorar la señal SIGINT (Ctrl-C).
- Deberá mostrar por pantalla las letras que ya fueron ingresadas.

Cliente:

- Solo podrá existir una instancia.
- Finalizará la ejecución ya sea por ganar o perder la partida no sin antes informar el resultado del juego. También puede finalizar si se recibe la señal SIGINT (Ctrl-C) en cuyo caso el servidor deberá dar por finalizada la partida.
- Deberá informar por pantalla al menos:
    - Cantidad de intentos restantes.
    - El estado de la palabra actual (letras descubiertas y ocultas).
    - En caso de perder la partida, develar la palabra en cuestión.
- Deberá proveer una forma de ingresar letras desde teclado.

Nota: el diseño de la interfaz del cliente queda a criterio del grupo, solo se pide respetar la información mínima que debe incluir.

Criterios de corrección:

Control	Criticidad
Compila sin errores con el makefile entregado	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Existe algún tipo de ayuda para la ejecución de los procesos (-h, --help)	Obligatorio
Utiliza semáforos y Memoria Compartida	Obligatorio
Elimina correctamente los recursos utilizados al finalizar	Obligatorio
Finalizan correctamente todos los procesos	Obligatorio
No hay pérdida de información	Obligatorio
Grafica el Hangman	Opcional

Ejercicio 5:

Realizar la implementación del juego Hangman con los mismos criterios pero generando una comunicación cliente-servidor mediante socket. El servidor debe recibir como parámetro el puerto que escuchara y el cliente recibirá la IP y puerto del servidor a donde deberá conectarse.

Criterios de corrección:

Control	Criticidad
Compila sin errores con el makefile entregado	Obligatorio
Funciona correctamente según enunciado	Obligatorio
Existe algún tipo de ayuda para la ejecución de los procesos (-h, --help)	Obligatorio
Genera una conexión TCP cliente-servidor	Obligatorio
Elimina correctamente los recursos utilizados al finalizar	Obligatorio
Finalizan correctamente todos los procesos	Obligatorio
No hay perdida de información	Obligatorio
Grafica el Hangman	Opcional

Actividad Práctica de Laboratorio 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:** Esta actividad práctica no es 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>
APL 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:

1/5/2009 ---------- listo
1/5/2010 ---------- listo
1/5/2011 ---------- listo
1/5/2012 ---------- listo
1/5/2013 ---------- listo
1/5/2014 ---------- listo
1/5/2015 ---------- listo
1/5/2016 ---------- listo
1/5/2017 ---------- listo
1/5/2018 ---------- listo
1/5/2019 ---------- 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 pen-drive.
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, 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)?