

EVALUACIÓN	Obligatorio 1	GRUPO	Mat. y Noct	FECHA	28-11-2024
MATERIA	Infraestructura				
CARRERA	Licenciatura en Sistemas				
CONDICIONES	<p>- Puntaje máximo: 35 puntos - Puntaje mínimo: 1 punto - Fecha de entrega: 28/11/2024 hasta las 21:00 horas en gestion.ort.edu.uy (max. 40Mb en formato zip, rar o pdf)</p> <p>Defensa Fecha de defensa: Las defensas serán coordinadas con el docente luego de la entrega.</p> <p><u>La defensa es obligatoria y eliminatoria.</u> El docente es quien definirá y comunicará la modalidad, y mecánica de defensa. <u>La no presentación a la misma implica la pérdida de la totalidad de los puntos del Obligatorio.</u></p> <p>Uso de material de apoyo y/o consulta</p> <p><u>Inteligencia Artificial Generativa</u></p> <ul style="list-style-type: none"> - Seguir las pautas de los docentes: Se deben seguir las instrucciones específicas de los docentes sobre cómo utilizar la IA en cada curso. - Citar correctamente las fuentes y usos de IA: Siempre que se utilice una herramienta de IA para generar contenido, se debe citar adecuadamente la fuente y la forma en que se utilizó. - Verificar el contenido generado por la IA: No todo el contenido generado por la IA es correcto o preciso. Es esencial que los estudiantes verifiquen la información antes de usarla. - Ser responsables con el uso de la IA: Conocer los riesgos y desafíos, como la creación de "alucinaciones", los peligros para la privacidad, las cuestiones de propiedad intelectual, los sesgos inherentes y la producción de contenido falso - En caso de existir dudas sobre la autoría, plagio o uso no atribuido de IAG, el docente tendrá la opción de convocar al equipo de obligatorio a una defensa específica e individual sobre el tema <p>IMPORTANTE:</p> <ol style="list-style-type: none"> 1) Inscribirse 2) Formar grupos de hasta 3 personas del mismo dictado 3) Subir el trabajo a Gestión antes de la hora indicada (ver hoja al final del documento: "RECORDATORIO") 4) Se habilitará un foro para realizar consultas del obligatorio. <p>Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador de cursos o Coordinación adjunta antes de las 20:00h del día de la entrega, a través de los mails crosa@ort.edu.uy / posada_l@ort.edu.uy (matutino) / larrosa@ort.edu.uy (nocturno), o vía Ms Teams.</p>				

Introducción

Este obligatorio tiene 4 partes, hacen foco a distintas tecnologías vistas en el curso. Una primera parte a resolver con Bash, una segunda con POSIX, una tercera con ADA y cuarta con Docker.

Parte 1 Bash:



El objetivo de esta tarea es que los estudiantes demuestren su comprensión y habilidades prácticas en el uso de los comandos principales de la línea de comandos de Linux mediante la creación y ejecución de un script de Bash. En esta tarea, los estudiantes diseñarán un script que simulará una serie de acciones en un sistema Linux, utilizando una variedad de comandos para manipular archivos y directorios, interactuar con el sistema y realizar tareas comunes de administración.

Ingreso

Se tienen dos documentos de texto, un documento que guarda usuarios y contraseñas de administradores y otro que guarda usuarios y contraseñas de clientes.

Inicialmente, se solicita registrar el usuario admin con la password admin.

Al iniciar el script se solicitan las credenciales (las cuales se verifican con el archivo que corresponda). De ser correcta la autenticación, se muestra el menú que corresponda.

Requerimientos:

El estudiante debe tener en cuenta el ingreso de usuarios vacíos para la autenticación.

Se penalizará el ingreso al menú si las credenciales no corresponden con su contraseña.

Acciones disponibles para los clientes

Iniciar sesión

El cliente debe poder iniciar sesión con su cédula y contraseña previamente registrada por el administrador.

Listar mascotas disponibles para adopción

Se debe mostrar el nombre, tipo, edad, descripción

Ejemplo: Rex - Perro - 5 - Juguetón, cariñoso y muy enérgico.

Adoptar mascota

Se listan las mascotas por número y por nombre. El cliente ingresa el número de la mascota que desea adoptar. Una vez confirmada la acción se elimina la mascota de las adopciones disponibles y se registra la adopción en un archivo adopciones.txt con los datos previos y la fecha de adopción en formato dd/mm/yyyy.

Salir

Acciones disponibles para los administradores

Registrar usuario

Se debe ingresar nombre, cédula, número de teléfono y fecha de nacimiento. Se debe verificar que el usuario no exista previamente.

Nota: El estudiante es libre de decidir cómo le consulta al usuario si las nuevas credenciales corresponden a un cliente o un administrador. Un usuario NO puede ser cliente y administrador a la vez.

Registro de mascotas

Se solicita al usuario ingresar los siguientes datos: numero Identificador, tipo de mascota, nombre, sexo, edad, descripción y fecha de ingreso al sistema.

Ejemplo: 001 - Perro - Rex - Macho - 5 - Juguetón - 15/03/2024

Requerimientos:

Numero Identificador, debe ser un número entero y no se debe repetir, no es necesario que este sea consecutivo, pueden quedar números salteados (en el caso de adopción se elimina el registro).

Edad, solo se admiten números enteros mayores a cero, se asume que todas las mascotas tienen más de un año.

Siempre que el usuario termina una acción debe volver al menú principal o en caso de salir a la autenticación.

Estadísticas de adopción

Se desea conocer el impacto que el refugio está generando en la comunidad, es por eso que se solicita generar un conjunto de estadísticas que ayuden a evaluar su desempeño.

Es necesario poder conocer el porcentaje de adopción para cada tipo de mascota.

Determinar en qué mes se realizan más adopciones.

Edad promedio de los animales adoptados.

Salir

Formato de entrega:

Se debe entregar una carpeta con todos los archivos adentro (incluidos los datos de prueba).

El estudiante debe conocer toda la solución, debe ser capaz de explicar la funcionalidad de cada línea ingresada. En caso de utilizar IA debe de conocer la totalidad de los comandos que la IA le suministra, y debe citar la misma.

Parte 2 Posix:

Precedencias con C:

Pedro es estudiante de Ingeniería en Sistemas. Tuvo dificultades para organizar sus estudios, por lo que decidió pedir ayuda a su consejero académico. Este le proporcionó el plan de materias con sus respectivas precedencias, que es el siguiente:

Introducción a la Programación: Sin previas.

Matemáticas I: Sin previas.

Física I: Sin previas.

Estructuras de Datos: Requiere Introducción a la Programación.

Matemáticas II: Requiere Matemáticas I.

Física II: Requiere Física I.

Programación Avanzada: Requiere Estructuras de Datos y Matemáticas II.

Bases de Datos: Requiere Estructuras de Datos.

Redes de Computadoras: Requiere Programación Avanzada y Física II.

Sistemas Operativos: Requiere Programación Avanzada y Redes de Computadoras.

Ingeniería de Software: Requiere Programación Avanzada.

Seguridad Informática: Requiere Redes de Computadoras y Bases de Datos.

Inteligencia Artificial: Requiere Programación Avanzada y Matemáticas II.

Computación Gráfica: Requiere Física II y Programación Avanzada.

Desarrollo Web: Requiere Bases de Datos y Redes de Computadoras.

Sistemas Distribuidos: Requiere Sistemas Operativos y Redes de Computadoras.

Big Data: Requiere Bases de Datos y Matemáticas II.

Robótica: Requiere Física II y Programación Avanzada.

Ciberseguridad: Requiere Seguridad Informática y Sistemas Operativos.

Análisis de Algoritmos: Requiere Programación Avanzada y Matemáticas II.

Para ayudar a Pedro, pedimos representar el grafo de precedencias, y realizar en base a este un programa en C que imprima el contenido del nodo, respetando las precedencias del grafo. Utiliza threads para las tareas que pueden hacerse en simultáneo.

Para facilitar la tarea, a continuación se encuentra una lista que muestra el código de cada materia. Se puede usar este para la representación del grafo y la construcción del programa.

Introducción a la Programación: IP

Matemáticas I: M1

Física I: F1

Estructuras de Datos: ED

Matemáticas II: M2

Física II: F2

Programación Avanzada: PA

Bases de Datos: BD

Redes de Computadoras: RC

Sistemas Operativos: SO

Ingeniería de Software: IS

Seguridad Informática: SI

Inteligencia Artificial: IA

Computación Gráfica: CG

Desarrollo Web: DW

Sistemas Distribuidos: SD

Big Data: BD

Robótica: RO

Ciberseguridad: CS

Análisis de Algoritmos: AA

Nota: Se debe subir el archivo .c y el grafo dibujado (digitalmente, no a mano) por el/los estudiantes.

Parte 3. ADA:

Se debe crear una simulación de una CPU basada en Little Man Computer (LMC). El LMC es un modelo didáctico de procesador, creado por el Dr. Stuart Madnick en 1965. El LMC se utiliza generalmente para enseñar, y modela un simple procesador de arquitectura von Neumann, que tiene (casi) todas las características básicas de un ordenador moderno.

Se puede programar en código máquina. También es llamada MÁQUINA TEÓRICA, MÁQUINA VIRTUAL, PROCESADOR ELEMENTAL. La versión que veremos tiene algunas modificaciones respecto a la original. Suponemos que implementaremos dos procesadores, el 0 y el 1, que trabajan sobre una misma memoria. La memoria tiene 128 registros donde se guardan números enteros o el tipo de dato que crea conveniente para resolver el problema. Va desde la dirección 0 hasta la 127.

Ambos procesadores pueden leer y escribir la memoria. También pueden leer y escribir en un teclado virtual y una consola virtual (que en realidad son la consola en que corremos la simulación). Existen hasta 16 semáforos en un array de semáforos (del 0 al 15). Estos responden a 3 operaciones. Con init recibe un parámetro que les inicializa su valor interno, luego wait y signal tiene el funcionamiento habitual en los semáforos. Los semáforos por defecto se inician en cero.

El acceso a cada registro de la memoria es exclusivo (tenga en cuenta que se pueden tener dos procesadores intentando acceder a la memoria a la vez). La memoria solo se lee o se escribe. Son las dos únicas operaciones que tiene.

Nuestro procesador tiene dos registros de programador (puede necesitar más para implementar operaciones). El registro A o acumulador, donde se hacen las sumas y restas, y se lee o se escribe desde y hacia la consola y el teclado. Y el puntero de instrucción IP que es el que lleva la cuenta de qué registro de memoria se lee para ejecutar la instrucción que corresponde.

Tenga en cuenta que el programa debe utilizar ambos CPUs, no será válido que todas las sentencias sean ejecutadas por un único procesador. Recomendamos realizar un modo “depuración” que indique cada sentencia en qué procesador se está ejecutando.

Las operaciones (que definen al programa) deben estar cargadas en memoria mediante una operación

1) MEM.WRITE([Direccion],[Valor])

Operaciones básicas

2) LOAD [Parámetro]

Carga al acumulador lo que se encuentra en la

3) STORE [Parámetro]

Operación inversa al LOAD, guarda a la dirección de memoria dada lo que se encuentra en el acumulador

4) ADD [Parámetro]

Suma al acumulador lo que se encuentra en la dirección dada en el parámetro, guarda el resultado en el acumulador.

5) SUB [Parámetro]

Resta al valor del acumulador, lo que se encuentra en la dirección dada por parámetro, guarda el resultado en el acumulador.

En la siguiente URL puede corroborarse el comportamiento del IP, y Ac para cada instrucción anterior <https://bit-machine.co.uk/machine.html>

Nota:

- Cada CPU podrá imprimir mediante STATUS, el valor de dichas variables.
- Cada CPU será una tarea, con sus propias variables
- Al aplicar una sentencia a la CPU su IP aumenta en 1.

Para nuestro ejercicio debemos agregar 4 instrucciones más a nuestro procesador. Estas instrucciones son:

6) BRCPU [Parametro] Saltar si el procesador es el número siguiente. Esto es similar a BRZ o BRP, pero en vez de verificar el valor del acumulador, comparan con un número identificador de cada CPU. Por lo demás es el mismo funcionamiento. Es decir, nos permite conocer si estamos en el procesador 1 o 2.

7) SEMINIT [Parametro], [Parametro] Inicializar el semáforo con el número siguiente al valor del segundo número. O sea que esta instrucción debe tener dos operandos (Operand). Una es el número de semáforo y la otra es el valor que se le asigna. Precisamos que al menos haya dos semáforos (0 y 1). Pero podría soportar más. Los semáforos se inicializan siempre a cero por defecto.

8) SEMWAIT [Parametro] Realizar un wait en el semáforo indicado por el número del operando. Pues la CPU deberá aplicar un wait al semáforo indicado. Y si es el caso se bloqueará. Sino continuará con su ejecución.

9) SEMSIGNAL [Parametro] Realizar un signal en el semáforo indicado por el número del operando. Entonces la CPU deberá aplicar un signal al semáforo indicado y continuará con su operación. Y si otra CPU estaba bloqueada, en este caso se desbloqueará y

continuará con su operación. A todos los efectos, el semáforo deberá comportarse como un semáforo genérico hecho en ADA.

Se deberá hacer un programa para nuestras CPUs que a una variable que llamaremos “valor” inicializada en 8, una CPU le suma 13, y la otra CPU le suma 27. Al final una de las dos CPUs debe devolver en consola (OUT) el resultado. Deben controlar la exclusión mutua y la sincronización de ambas CPUs para que dé siempre resultados consistentes.

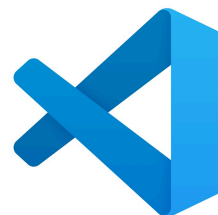
Se permite agregar sentencias que crea conveniente, siempre que detalle correctamente su funcionalidad.

Realizar un programa en ada que resuelva el problema anterior, adjuntar documentación explicativa de la solución elegida.

Parte 4. Docker

Parte 4.1 - Investigación

Usted es parte del equipo de Infra en su empresa, como parte del desarrollo de una IDP (Internal Developer Platform) se le solicita configurar un entorno de desarrollo sencillo para que cualquier developer pueda realizar modificaciones sobre un código dado de forma sencilla y accesible. Uno de los requerimientos que le imponen es que debe ser agnóstica a la plataforma de uso, debe ser accesible desde un navegador web y en lo posible debe consumir la menor cantidad de recursos. Ya que domina de forma excepcional todos los conocimientos de Docker, Contenedores y virtualización, se le ocurre la idea de realizar una instalación de **Visual Studio Code Server** en un **contenedor de Docker**.



Se debe investigar:

- Que es una IDP
 - Usos de IDP
- Para crear la imagen:
 - Dockerfile:
 - ¿Qué requerimientos hay?
 - ¿Qué imagen usar de base?
 - ¿Qué puertos se debe exponer para el acceso?
 - ¿Archivo de configuración?
- Para ejecutar el code server
 - Docker compose (estructura)
 - Servicio, puertos y volúmenes consideran necesarios

Parte 4.2 - Parte Práctica

En el mundo actual, las bases de datos son esenciales para almacenar y procesar grandes cantidades de información. Una de las bases de datos más utilizadas es SQL Server, que ofrece una amplia gama de características y es compatible con una gran variedad de aplicaciones. Para facilitar la instalación y administración de SQL Server, se puede utilizar Docker. En este ejercicio, exploraremos cómo **levantar una base de datos SQL Server** utilizando Docker y cómo conectarse a ella mediante herramientas como DBeaver o Azure Data Studio. También veremos cómo utilizar Compose para ejecutar la base de datos de forma más eficiente.




- Comando de docker para levantar una base de datos SQL server
- Conexión con [DBeaver](#) o [Azure Data Studio](#)
- Compose para ejecución

• RECORDATORIO: IMPORTANTE PARA LA ENTREGA

Obligatorios

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem “Evaluaciones” y la instancia de evaluación correspondiente, que figura bajo el título “Inscripto”.
3. Para iniciar la entrega hacé clic en el ícono: 
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé clic en “Agregar”. El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hacé clic en “Crear equipo”.

Cualquier integrante podrá:

- **Modificar la integración del equipo.**

- **Subir el archivo de la entrega.**

1. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en “Subir” para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

Cuando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.

9. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
10. Podés cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
11. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
12. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
13. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador de cursos o Coordinación adjunta antes de las 20:00h del día de la entrega, a través de los mails crosa@ort.edu.uy / posada_l@ort.edu.uy (matutino) / larrosa@ort.edu.uy (nocturno), o vía Ms Teams.