TRABAJO PRÁCTICO Nº 2

SISTEMAS OPERATIVOS DISTRIBUIDOS

Temas de la unidad:

1. Modelo Cliente Servidor
   a. Comunicación entre procesos
   b. RPC
   c. Memoria compartida
2. Clusters
   a. Tipos de interconexión
   b. Interfaz de red
   c. Planificación
      i) Algoritmos determinísticos
      ii) Algoritmos heurísticos
3. Gestión de procesos distribuidos
   a. Migración
   b. Sincronización
      i) Sección crítica distribuida
      ii) Interbloqueo distribuido
4. Sistemas de archivos distribuidos
   a. Nombrado
   b. Acceso remoto
   c. Semántica de consistencia
   d. Replicación
5. Modelo de Procesamiento basado en la Web
   a. Cliente Servidor múltiples capas (Middleware)
   b. Grid

Objetivos específicos:

- Que el alumno adquiera los conceptos básicos de los S.O. Distribuidos, sus distintos tipos y áreas de aplicación.
- Desarrollar en el alumno las habilidades necesarias para la instalación y administración de los S.O.A. Distribuidos.

PARTE 1

1) Escribir un script en bash que reciba como parámetro el nombre de un programa y responda si dicho programa está en ejecución o no.

2) Escribir un script en bash que reciba como parámetro el nombre de host y responda su MAC address.

3) Escribir un script en bash que informe la MAC address de la placa de red eth0.

4) Escribir un script en bash que informe los accesos al sistema en los últimos 3 días.

5) Escribir un script en bash que monitorice un servicio recibido como parámetro. Cuando se detecte que dicho servicio está inactivo, enviar un email al administrador reportando dicha inactividad.

6) Escribir un script en bash que monitorice la memoria RAM. Cuando se detecte que la memoria disponible sea menor que 500mb, enviar un email al administrador reportando dicha inactividad.

7) Escribir un script en bash que monitorice un enlace. Si no se reciben de dicho enlace por lo menos 3 paquetes cada 2 segundos, con una espera máxima de 3 segundos, mostrar un mensaje indicando que el enlace está inactivo.

PARTE 2

8) Escribir un script en lenguaje C que realice una system call "*clear*".

9) Describir similitudes y diferencias entre los entornos de desarrollo MPI y OpenMP.

10) Para aplicaciones que utilizan una librería MPI (MPICH/OpenMPI), describa:
- **a.** Cómo instalar el entorno de desarrollo.
- **b.** Cómo compilar.
- **c.** Cómo ejecutar.

11) Para aplicaciones que utilizan OpenMP, describa:
- **a.** Cómo instalar el entorno de desarrollo.
- **b.** Cómo compilar.
- **c.** Cómo ejecutar.

12) ¿Para qué sirven las siguientes funciones de MPI?
- **a.** MPI_Init(…)
- **b.** MPI_Comm_rank(…)
- **c.** MPI_Comm_size(…)
- **d.** MPI_Send(…)
- **e.** MPI_Recv(…)
- **f.** MPI_Finalize(…)

13) ¿Para qué sirven las siguientes directivas al compilador de OpenMP?
- **a.** #pragma omp parallel
- **b.** #pragma omp parallel private(a)
- **c.** #pragma omp parallel shared(a)
- **d.** #pragma omp parallel for
- **e.** omp_get_thread_num()
- **f.** omp_get_num_threads()

14) ¿Cómo se puede implementar una sección crítica…
- **a.** con MPI?
- **b.** Con OpenMP?

15) Nombrar 3 File Systems distribuidos y describir sus principales diferencias.

16) Dado el siguiente código en lenguaje C para encontrar números primos hasta el N, tanto de forma secuencial como paralela con y sin memoria compartida. Sacar conclusiones sobre el rendimiento de cada uno de ellos utilizando *"/usr/bin/time*" y "*perf*".

https://github.com/utnfrlp/SOA/blob/master/TP2/primos.c
https://github.com/utnfrlp/SOA/blob/master/TP2/primos_mpi.c
https://github.com/utnfrlp/SOA/blob/master/TP2/ejemplo_OpenMP.c

17) Dado el siguiente código en lenguaje C:
https://github.com/utnfrlp/SOA/blob/master/TP2/ejemplo_MPI.c
- **a.** Agregue una nueva sentencia *printf* luego de la sentencia *printf* existente.
- **b.** Compile utilizando MPI y ejecute en paralelo.
- **c.** Sincronice la ejecución de los procesos utilizando Barreras de memoria mediante la función **MPI_Barrier(MPI_COMM_WORLD)**.
- **d.** Ejecute 10 veces el programa sincronizado. ¿Qué sucede con la sincronización de los procesos?

18) Dado el siguiente código en lenguaje C:
https://github.com/utnfrlp/SOA/blob/master/TP2/ejemplo_OpenMP.c
- **a.** Agregue una nueva sentencia *printf* luego de la sentencia *printf* existente.
- **b.** Compile utilizando OpenMP y ejecute en paralelo.
- **c.** Sincronice la ejecución de los procesos utilizando Barreras de memoria mediante la función **#pragma omp barrier**.
- **d.** Ejecute 10 veces el programa sincronizado. ¿Qué sucede con la sincronización de los procesos?

19) Escribir un programa en C que reciba como parámetro un valor X y de manera distribuida realice los siguientes cómputos:
- **a.** En proceso 0: $Y=X^2$
- **b.** En proceso 1: $Y=X^3$
- **c.** En proceso 2: $Y=X^4$

20) Dado el siguiente enunciado: "Sea un array de números enteros, correspondiente al precio de 1 millón de productos, se necesita realizar un incremento de un 20% en el precio de cada uno de ellos":
- **a.** Implementar un script secuencial en lenguaje C que resuelva el problema.
- **b.** Modificar el script del punto anterior para que la ejecución se realice de forma paralela sobre los elementos del array.
- **c.** Sacar conclusiones de rendimiento utilizando el comando "*/usr/bin/time*", para distintas cantidades de subprocesos/threads generados.

21) ***(EXTRA)*** Utilizando la librería *mpi4py*
https://mpi4py.readthedocs.io/en/stable/tutorial.html, reescribir los *ejercicios 19 y 20* a lenguaje *Python*.

- *Condiciones de entrega: se debe entregar un fichero .zip con todos los ejercicios dentro, el cual deberá tener el nombre del grupo y el número de trabajo práctico, por ejemplo:* ***GRUPO_01_TP2.zip***. Dicho fichero será subido a Moodle por un integrante del grupo y, para que la entrega se haga efectiva, deberá ser validado por los demás integrantes.

- *Condiciones de evaluación: Como pueden existir diversas formas de realizar cada ejercicio, siempre que se consiga el objetivo del mismo se considerará que la respuesta es válida.*

- *Entregas fuera de los plazos estipulados de entrega tendrán impacto en la nota, pudiendo hacer que el trabajo práctico sea desaprobado.*