UNIVERSIDAD AUTONOMA METROPOLITANA

Programación Avanzada

Práctica 3:

"Algoritmos iterativos de Búsqueda"

(Comparación de desempeño)

Fecha de Elaboración:
Martes, 8 de julio del 2008

Fecha de Entrega:
Martes, 15 de julio del 2008

Contenido

1 Objetivos 2
2 Introducción 3
2.1 Complementos 3
2.2 Ayuda complementaria 3
3 Desarrollo 4
3.1 Actividad 1 "Comandos en el Entorno de Linux (2)" 4
3.2 Actividad 2 "Algoritmos Iterativos de Búsqueda" 7
3.2.1 Búsqueda Secuencial 7
3.2.2 Diseño Modular Búsqueda Secuencial 7
3.2.3 Búsqueda Binaria 7
3.2.4 Diseño Modular Búsqueda Binaria 7
3.2.5 Datos de Prueba de los Programas de Ordenamiento. 7
3.3 Actividad 3 "Algoritmos Iterativos de Búsqueda – Datos desde un archivo y evaluación de desempeño" 8
3.3.1 Parte 1 "Medición de tiempo de ejecución con funciones de la librería time.h y con el comando time de Linux" 8
3.3.2 Parte 2 "Extracción de datos desde archivos." 8
3.3.3 Diseño Modular Búsqueda Secuencial Archivo 10
3.3.4 Diseño Modular Búsqueda Binaria 10
3.3.5 Reporte de la actividad. 11
3.4 Actividad 4 "Implementación del Programa 1 (Opcional)" 11

AVISOS:

- No se reciben reportes de las prácticas después de la fecha de entrega.
- En la página se encuentra el formato del reporte para las prácticas en donde se describen cada uno de los apartados que deberá considerarse. Se deberá seguir dicha plantilla para el desarrollo del reporte de las prácticas.
- Enviar a la cuenta de correo un paquete zipeado que contendrá:
  - Códigos fuentes (extensión **.c**).
  - Códigos Objetos (extensión **.o**)
  - Ejecutables (sin extensión)
  - En el reporte incluir por actividad (extensión **.doc)**:
    - **Desarrollo** de la actividad.
    - **Resultados** de la actividad.
- En el subject del correo escribir: ***Práctica 3 - <nombre completo>***.

1 Objetivos

1) Utilizar **comandos** dentro del entorno de Linux para facilitar las fases de codificación, compilación, ejecución y pruebas, en el desarrollo de programas.
2) Utilizar el diseño de programación ascendente y descendente en el enfoque de **programación modular** (enfoque "divide y vencerás") para la resolución de problemas mediante el uso de **módulos** que serán traducidos a funciones (y/o procedimientos) en el lenguaje C.
3) Implementar los **algoritmos de búsqueda**: "Secuencial" y "Binaria".
4) Integrar un mecanismo de medición de tiempo para determinar el **desempeño de los algoritmos** con distintos datos, con el objetivo de determinar las diferencia del tiempo de respuesta de cada uno de los algoritmos de ordenamiento.
5) Integra módulos que permitan la extracción de los datos desde un **archivo** de texto.

2 Introducción

En el desarrollo de esta práctica se requiere que el alumno realice cada una de las actividades que se le indica.

En las actividades referentes al entorno Linux, el alumno ingresara en la línea de comandos algunas instrucciones y observará los resultados generados.

La implementación de los algoritmos de búsqueda se requiere la traducción de los algoritmos de pseudocódigo a código en C.

2.1 Complementos

Para completar el reporte de la práctica el alumno deberá de efectuar una breve investigación de los siguientes puntos (máximo 3 párrafos por punto):

- Investigar el orden de complejidad de los algoritmos de iterativos de búsqueda.
- Apertura, lectura, escritura y cierre, de archivos de texto y binarios.
- Lectura y escritura de información con formato utilizando las funciones: **fscanf** y **fprintf**, respectivamente.

2.2 Ayuda complementaria

- Referencia de la bibliografía contenida en la planeación del curso de Programación Avanzada.
- Notas del Curso de Introducción a al Programación 08-I.
- Guía Básica para el uso de C.

3 Desarrollo

La sección del desarrollo de la práctica se divide en una serie de actividades que permiten englobar, desarrollar y alcanzar cada uno de los objetivos de la práctica. Es necesario leer con atención cada una de las actividades a desarrollar.

3.1 Actividad 1 “Comandos en el Entorno de Linux (2)”

Con el desarrollo de esta actividad se alcanzará el objetivo 1 de la práctica.

Para el desarrollo de esta actividad siga en orden los siguientes pasos, verifique que los resultados son los mismos que se mencionan a continuación:

1) Entrar al sistema escribiendo el nombre de usuario (login) y contraseña (password).
2) Abrir una ventana de “shell” (Terminal) para practicar varios de los comandos de línea permitidos por el sistema.

3) Practicar los siguientes comandos y observe los resultados:

- **grep**
  Muestra la línea en donde se encuentra una cadena dentro de un archivo.

```
{~/Practica3}[pa@quetzal21]$ cal > archivo.txt

{~/Practica3}[pa@quetzal21]$ more archivo.txt
    julio 2008
lu ma mi ju vi sÃ¡ do
   1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31


{~/Practica3}[pa@quetzal21]$ grep "11" archivo.txt
 7  8  9 10 11 12 13

{~/Practica3}[pa@quetzal21]$ grep "31" archivo.txt
28 29 30 31
```

- **su**
  Permite cambiar cambiarse temporalmente a la cuenta de otro usuario. Si no se especifica el nombre del usuario, se considera que se desea pasar a la cuenta del root (usuario administrador del sistema). En ambos casos se pedirá la contraseña.

```
{~}[gustavob@quetzal21]$ su pa
ContraseÃ±a:
```

{**/lab/profesores/gustavob**}[**pa**@quetzal21]$ **whoami**
pa

{**/lab/profesores/gustavob**}[**pa**@quetzal21]$ **ls**
ls: no se puede abrir el directorio .: Permiso denegado

{/lab/profesores/gustavob}[pa@quetzal21]$ **cd**

{**~**}[pa@quetzal21]$ **ls**
1 converetemperatura.c DAJ

{~}[pa@quetzal21]$ **exit**
exit

{~}[gustavob@quetzal21]$ **whoami**
gustavob

- **ps**

Permite ver los procesos que se están ejecutando en el sistema.
Este comando es util para determinar si un programa se encuentra en un ciclo infinito.
Para ver solo los procesos principales.

{~}[gustavob@quetzal21]$ **ps**

```
  PID TTY          TIME CMD
 6341 pts/2    00:00:00 bash
 6510 pts/2    00:00:00 ps
```

El **PID** es el identificador del proceso.

Para ver solo los procesos principales de un usuario.

{~}[gustavob@quetzal21]$ **ps -u**
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
gustavob  6341  0.0  0.6   5668  3140 pts/2    Ss   09:56   0:00 -bash
gustavob  6518  0.0  0.2   2644  1072 pts/2    R+   10:08   0:00 ps -u
```

Para ver todos los procesos del sistema.
{~}[gustavob@quetzal21]$ **ps -e**
{~}[gustavob@quetzal21]$ **ps -ea**

Para ver todos los procesos del sistema incluyendo el usuario
{~}[gustavob@quetzal21]$ **ps -eua**

Experimento

Abra otra terminar y dentro de esa terminar ejecute:
{~}[gustavob@quetzal21]$ grep "hola"

En la otra terminar ejecute y busque el poceso:
{~/Practica1}[pa@quetzal21]$ ps –aue
gustavob 6532 0.0 0.1 3016 768 pts/2 S+ 10:13 0:00 grep hola

No olvide terminar el proceso con <ctrl+c>

- **mesg**

  Habilita o inhabilita la entrada de mensaje.

  {~}[gustavob@quetzal21]$ **mesg**
  is **y**
  {~}[gustavob@quetzal21]$ **mesg n**
  {~}[gustavob@quetzal21]$ **mesg**
  is n

- **write**

  Envía una nota a otro usuario del sistema. Para terminar de escribir el mensaje, teclee <Ctrl.+D>

  Terminal 1:

  {~}[gustavob@quetzal21]$ **write pa**
  Hola como esta la practica3?

  Terminal 2:

  Message from gustavob@quetzal21 on pts/2 at 10:17 ...
  Hola como esta la practica3?
  EOF

3.2 Actividad 2 "Algoritmos Iterativos de Búsqueda"

Con el desarrollo de esta actividad se alcanzará los objetivos 2 y 3 de la práctica.

3.2.1 Búsqueda Secuencial

Efectué la implementación del algoritmo de Búsqueda Secuencial (**BSecuencial.c**). Desarrolle el diagrama de modular correspondiente.

Entrada: Numero de elementos a ingresar, los valores enteros que se almacenarán en un arreglo. El valor a buscar dentro del arreglo con el algoritmo de búsqueda secuencial.

Salida: Un mensaje indicando en que posición del arreglo se encuentra el elemento ingresado o en caso contrario un mensaje indicando que no se encontró dicho elemento. Se debe de indicar el tiempo de ejecución.

3.2.2 Diseño Modular Búsqueda Secuencial

3.2.3 Búsqueda Binaria

Efectué la implementación del algoritmo de Búsqueda Binaria (**BBinaria.c**) para un arreglo previamente ordenado con uno de los tres algoritmos desarrollado en la práctica 2. Desarrolle el diagrama modular correspondiente

Entrada: Numero de elementos a ingresar y los números enteros que se almacenarán en un arreglo.

Salida: El arreglo ordenado paso por paso hasta obtener el arreglo ordenado con el método de inserción directa y el tiempo de ejecución.

3.2.4 Diseño Modular Búsqueda Binaria

3.2.5 Datos de Prueba de los Programas de Ordenamiento.

Orden ascendente: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,14, 15, 16, 17, 18, 19, 20.
Orden descendente: 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.
Pseudoaleatorio: 3, 10, 9, 1, 8, 2, 7, 5, 4, 6, 20, 11, 19, 12, 18, 13, 17, 14, 15,16.

3.3 Actividad 3 "Algoritmos Iterativos de Búsqueda – Datos desde un archivo y evaluación de desempeño"

Con el desarrollo de esta actividad se alcanzará los objetivos 2, 3, 4 y 5 de la práctica.

3.3.1 Parte 1 "Medición de tiempo de ejecución con funciones de la librería time.h y con el comando time de Linux"

Modificar los dos programa obtenido en la actividad 2 (**BSecuencialArch.c** y **BBinariaArch.c**) para que tome el tiempo de ejecución del cuerpo de la función de búsqueda, utilizando las funciones de la librería **time.h**.:

Al ejecutar el programa utilice también el comando **time** de Linux para comparar resultados.

Con dichas modificaciones verifique el funcionamiento del programa antes de continuar con la siguiente parte de la actividad.

3.3.2 Parte 2 "Extracción de datos desde archivos."

Antecedentes de Archivos (*extraído de los apuntes del curso de introducción a la programación*).

Definición 17: Estructura de declaración y apertura de un tipo de dato **Archivo** (**FILE**):

Lenguaje C

```
//Declaración de un tipo de dato Archivo
FILE *archivo;
//Apertura de una Archivo
//        fopen (nombre_archivo, modo_apertura)
fopen ("C:\\arch.txt", "r");
```

Ejemplo:

```
char ruta_nombre[] = "C:\\PWS.txt";
//Apertura del Archivo
arch = fopen(ruta_nombre, "r");

//Verificación de apertura
if(arch == NULL )
{
    printf("\n\nError al abrir el archivo %s ",
  ruta_nombre);
    exit( -1);  //Finalizar el programa
}
else //No es necesario 'otro caso'
{
    printf("\n\nOK archivo %s abierto!!!!", ruta_nombre);
}
```

Definición 18: Modos de apertura de un archivo:

Modo	Significado
"r"	Abre el archivo para ***lectura***.
"w"	Crea y abre un nuevo archivo, si ya existía se ***pierde*** la información.

"a"	Abre el archivo ***existente*** para escribir al **final**.
"r+"	Abre archivo ya existente para **leer o escribir** la información.
"w+"	Crea un archivo para **leer o escribir**, si ya existía se ***pierde*** la información.
"a+"	Abre el archivo ***existente*** para **leer o escribir** al **final**.

Los modos de archivo para abrir un archivo que se muestran en la definición 18, son utilizados para abrir texto por defecto; sin embargo es posible utilizar el carácter 't' o 'b' para indicar que se trata de un archivo de texto o un archivo binario, respectivamente.

- Modos para apertura de un archivo de ***texto***:
  "rt", "wt", "at", "r+t", "w+t", "a+t"
- Modos para apertura de un archivo ***binario***:
  "rb", "wb", "ab", "r+b", "w+b", "a+b"

Definición 19: Cierre de un Archivo:

Lenguaje C

```
//Cierre de una Archivo
//          fclose (identificador_del_archivo)
fclose (archivo);
```

Cuando se esta efectuando la lectura de la información de un archivo el compilador requiere de un indicador que le informe que se ha llegado al final del archivo y que ya no hay más datos a leer, este indicador es conocido como la condición de ***Fin De Archivo*** (**End Of File - EOF**).

Definición: Lectura de caracteres desde un Archivo utilizando *fscanf*():

Lenguaje C

```
//Declaración de un tipo de dato Archivo
FILE  *archivo;
int valor;
//Apertura…..

//Lectura continua
while ( !feof (archivo))   //mientras no se llegue al final del archivo
  {
          fscanf(archivo, "%d", &valor);
          printf("\n  Valor es: %d ",valor);
  }
```

Definición: Escritura de caracteres desde un Archivo utilizando *fprintf()*:

Lenguaje C

```
//Declaración de un tipo de dato Archivo
FILE  *archivo;
int valor = 10;
float  real = 20.3;
//Apertura…..
fprintf(archivo, "Esto aparece en el archivo: %d", valor);
fprintf(archivo ,"Dato Real: %f", real);
fprintf(archivo ,"%d-%f", valor, real);
```

Desarrollo de la Parte 2

Desarrollar un programa, que al momento de invocar su ejecución, reciba como argumento una cadena que representa la ruta, nombre y extensión del archivo que contiene los datos de prueba. En caso de que no se reciba dicho parámetro, solicitar el nombre dentro del programa.

Modifique los diseños modulares y agregue los módulos necesarios para dar soporte al acceso de los datos desde un archivo.

3.3.3 Diseño Modular Búsqueda Secuencial Archivo

3.3.4 Diseño Modular Búsqueda Binaria

3.3.5 Reporte de la actividad.

Los siguientes puntos se deben de considerar dentro de código fuente para obtenerlos como resultado de la ejecución del programa:

- Extracción exitosa de la información desde un archivo.
- Desarrollar la búsqueda de un elemento utilizando los algoritmos iterativos.
- Determinar el tiempo de ejecución con librería time.h y el comando time de Linux.

Con los resultados obtenidos para cada algoritmo respecto a un tipo de dato de prueba, efectué una tabla comparativa, en donde incluya sus conclusiones.

Datos de Prueba \ Algoritmo	**Archivo1.txt**	**Archivo2.txt**	**Archivo3.txt**
Secuencial (1)			
Binario (1)			
Secuencial (2)			
Binario (2)			

Tabla 1.- Comparativa de los algoritmos de búsqueda iterativo.

(1) Indica que el dato a buscar si se encuentra después de la **mitad** del arreglo.

(2) Indica que el dato a buscar **NO** se encuentra dentro del arreglo.

3.4 Actividad 4 “Implementación del Programa 1 (Opcional)”

Continué con la implementación del Programa 1, definiendo la estructura de datos Alumno, así como las funciones que fueron proporcionadas en clase.