Programación II. I.T. Informática de Sistemas y Gestión

Enunciado de la práctica

Curso 2009–10

1. Enunciado del problema

El sistema informático de la escuela almacena diversos registros sobre sus usuarios y actividad. Así por ejemplo `jair` dispone de un fichero `/var/log/apache2/access_log` con información, sobre los accesos que se han realizado al servidor web instalado en este equipo.

Las lineas del fichero `access_log` están divididas en campos con el siguiente significado:

- La dirección del visitante
- Un campo que permite una identificación más detallada del visitante. Cuando no se usa aparece en su lugar el símbolo –
- Si es distinto a – indica el *login* utilizado para acceder a un contenido protegido por *password*
- Fecha, hora y zona horaria del acceso
- Solicitud del visitante
- Código de estado HTTP devuelto por el servidor
- Número de bytes que envió el servidor, excluídas las cabeceras

En cuanto a los códigos de estado del servidor, se trata de números de tres cifras con el siguiente significado:

- `1??` Respuestas informativas
- `2??` Peticiones correctas
- `3??` Redirecciones
- `4??` Errores del cliente
- `5??` Errores de servidor

Ahora la escuela desea analizar los datos que proporciona ese fichero log.

1.1. Fase1

Elaborar un conjunto de clases que modelen la información disponible sobre los accesos al servidor web. Entre estas clases deberán estar obligatoriamente:

- `PETICION` Modela una petición `GET`, realizada sobre la web de un usuario de jair (las peticiones sobre la web del usuario «usuario» son peticiones sobre ficheros cuyo nombre comienza por `/~usuario` o `/%7Eusuario`). Dispone al menos de información sobre el usuario propietario de la web, el visitante que solicitó la descarga, el momento en que se solicitó, el resultado de la petición y el tamaño de la misma.
- `VISITANTE` Modela una máquina desde la que se realizan peticiones sobre webs de los usuarios de `jair`. Dispone al menos de información sobre la dirección de la máquina, peticiones realizadas, usuarios sobre los que ha realizado peticiones y tamaño total de los datos que ha recibido.

- *USUARIO* Modela un usuario sobre cuya web se han realizado peticiones. Dispone al menos de información sobre el nombre del usuario, las peticiones que se han realizado sobre él, los visitantes que ha recibido y el tamaño total de los datos que ha proporcionado.
- *LOG* Modela uno de estos ficheros log. Dispone al menos de una lista de peticiones, una de visitantes y otra de usuarios, además de acumular el volumen de datos descargados, número de peticiones correctas, erróneas, redirecciones,...

A la hora de elaborar estas clases debemos considerar que la dirección de la escuela cambia con frecuencia los datos que desea analizar, de modo que es importante que el diseño de las mismas sea tan flexible como sea posible.

1.2. Fase2

Elaborar una aplicación elemental de análisis de los datos del servidor. La aplicación recibirá como argumentos el nombre del fichero log a analizar y un usuario, devolviendo por pantalla una salida como la siguiente:

```
Resumen de las peticiones recibidas por el usuario <usuario>:
   <direccion_visitante> <codigo_http> <tamaño>
   <direccion_visitante> <codigo_http> <tamaño>
   ....
Total: <m> bytes en <n> peticiones.
```

La clase lanzadora de esta aplicación debe llamarse *FASE2*.

1.3. Fase3

Elaborar una aplicación de gestión de los datos del servidor que proporcione información más completa que en la fase anterior. La ayuda de esta aplicación, que funcionará mediante opciones de la línea de comandos, debe ser como la siguiente:

```
   --help                          Muestra esta ayuda
   --log <fichero>                 Hace el análisis sobre el fichero
                                   indicado en lugar de sobre ./access_log
   --resumen                       Información de resumen, con total de
                                   usuarios, visitantes y peticiones, así como
                                   porcentajes de peticiones con cada tipo de
                                   resultado. Además informa sobre el
                                   visitante que más peticiones ha realizado y
                                   el usuario que más peticiones ha recibido.
   --usuario <usuario>             Resumen de las peticiones recibidas por
                                   el usuario <usuario> con número total de
                                   peticiones y tamaño total de las mismas
   --visitante <visitante>         Información sobre los usuarios sobre los que
                                   ha realizado peticiones el visitante, con
                                   número total de peticiones realizadas y
                                   tamaño de las mismas
```

La clase lanzadora de esta aplicación debe llamarse *FASE3*.

2. Condiciones de entrega y calificación de la práctica

Para realizar la entrega, cada alumno deberá crear en *jair* un directorio *$HOME/pr2_0910* en el que almacenará los ficheros *.e* que desea entregar y un fichero *mi_correo* con una dirección de correo elec-

trónico para la confirmación de la entrega. Las clases lanzadoras de las distintas fases deberán llamarse `fase2.e` y `fase3.e` respectivamente.

Tanto el directorio de entrega como los ficheros deben ser accesibles para el usuario `felix`. El sistema recogerá exclusivamente los ficheros con extensión `.e` almacenados en ese directorio, sin analizar subdirectorios u otros ficheros, salvo `mi_correo`.

Las prácticas que no se ajusten estrictamente[1] a estas condiciones de entrega, o que no compilen, serán consideradas como no presentadas.

Aunque las prácticas tienen carácter individual, se admite el trabajo en parejas. Todo grupo de tres o más prácticas iguales será considerado fraudulento. En este sentido, el artículo 11.5 del Reglamento de Ordenación Académica determina que la realización fraudulenta de alguno de los ejercicios o trabajos exigidos para la evaluación de una asignatura supondrá una nota de cero, suspenso, en la convocatoria.

Las puntuaciones máximas de cada una de las fases de la práctica son:

- Fase1 + Fase2: 0,75 puntos
- Fase3: 0,25 puntos

3. Fechas límite de entrega

La práctica será recogida el día del examen de la asignatura a partir de las ocho de la tarde.

4. La práctica en la convocatoria extraordinaria

Esta práctica es válida para la convocatoria extraordinaria, en las mismas condiciones y con límite de entrega en el día del examen extraordinario de la asignatura. Si en esta convocatoria no se entrega una nueva práctica, la nota considerada será la obtenida en la convocatoria ordinaria.

5. Preguntas frecuentes sobre el enunciado

- **Si hemos hecho la práctica entre dos, ¿Tenemos que hacer dos entregas?** Efectivamente, cada uno de los alumnos de la pareja debe hacer su propia entrega si desea ser evaluado.
- **¿Y si no quiero que me notifiques la recogida?** No leas el correo. Si tu directorio no tiene un fichero `mi_correo` entenderemos que no deseas entregar la práctica.
- **¿Cómo hago accesibles los ficheros entregados al usuario `felix`?** Puedes utilizar el comando `setfacl`, en concreto `setfacl -m u:felix:rx $HOME/pr2_0910` nos da acceso al directorio de entregas mientras que `setfacl -m u:felix:r $HOME/pr2_0910/*` hace lo mismo con los ficheros que contiene en ese momento. Si permites que cualquier otro usuario copie o modifique tus ficheros, la responsabilidad es tuya.
- **¿Cómo puedo asegurarme de que mis ficheros ya son accesibles?** Puedes usar el comando `getfacl`.
- **Al leer un fichero me pasan «cosas raras»** Antes de nada debieras revisar el bucle de lectura de fichero. En SmartEiffel, como en muchos otros lenguajes, el final de fichero sólo es detectado después de una lectura, de modo que el esquema general para leer un fichero expresado en seudocódigo tiene un aspecto como el siguiente:

```
leer dato
mientras no final de fichero hacer
      procesar dato
      leer dato
fin_mientras
```

[1] Atención a las minúsculas en los nombres de los ficheros

- **¿Cómo puedo acceder a los argumentos de la linea de comandos en Eiffel?** Tienes un ejemplo en el fichero, `print_arguments.e`, del directorio `tutorial` de la instalación de SmartEiffel.
- **¿Cuál será mi puntuación si sólo entrego la fase 1?** Cero.
- **En la fase 3 no se especifica un fichero log por defecto ¿cómo identifico cuál es el fichero de log?** Echa otro vistazo al enunciado. Si no te dan un nombre de fichero, tienes que procesar el fichero que se llame `access_log` en el directorio actual.
- **Qué tiene que hacer la fase 3 si no le dan parámetros?** Lanzar un mensaje de error y la ayuda sobre el uso de la aplicación.
- **¿Es muy lento mi programa?** Probablemente, pero para evaluar la velocidad del ejecutable hay que tener en cuenta los parámetros de compilación utilizados.

  SmartEiffel utiliza por defecto el parámetro `-all_check` lo que provoca un ejecutable lento que incorpora todos los contratos y es ideal para las tareas de depuración. El parámetro `-boost` por el contrario hace que el ejecutable carezca de contratos, pero incorpore diversas optimizaciones al código, lo que lo hace mucho mas rápido.

  En general los sistemas UNIX realizan mediciones del tiempo consumido por un programa mediante el comando `time «programa»`.

  De todas formas si estás haciendo pruebas, quizás prefieras trabajar con contratos sobre una versión «más reducida» del fichero a analizar.