Manual de uso de `sigue`

Manual de uso de `sigue`

`sigue` es un conjunto de formatos y programas de dominio público que permiten registrar progreso de estudiantes y generar reportes periódicos. Aunque busca ser general, es particularmente apropiado para colegios colombianos, en los cuales permite mantener: plan de estudios, información de los estudiantes, información de los cursos, valoraciones de los estudiantes en diversas materias en diversos años; y permite imprimir reportes periódicos y finales para padres de familia. Cuenta con una interfaz en PHP para facilitar a profesores edición de logros, indicadores y valoraciones.

Dedicado

Este proyecto se dedica y encomienda al Dios único y Todopoderoso.

Tabla de contenidos

Lista de tablas

Capítulo 1. Introducción

1.1. Sobre `sigue`

`sigue` es un conjunto de formatos y programas que permiten registrar el progreso de estudiantes y generar reportes periódicos.

1.1.1. Características

- Adherencia a la legislación educativa vigente en Colombia (Ley 115/1994, Decreto 1860/1994 y Decreto 230/2002 – ver [ArchLeg]), pero también alta flexibilidad que permite adaptarlo en cada colegio colombiano de acuerdo al PEI de la comunidad educativa, e incluso en otros tipos de educación y eventualmente de otros paises.
- Permite mantener información de estudiantes, profesores así como planes de estudios y valoraciones de varios años en archivos XML. Por ejemplo para permitir que indicadores pendientes de un año pasen a otro, o para generar certificados de años anteriores. Como respaldo y para ayudar a organizar la información en el tiempo sugerimos el uso de CVS.
- Busca aprovechar al máximo la información disponible sobre estudiantes y profesores en una LAN con computadores tipo Unix (por ejemplo como la descrita en [AALinux]). Sin embargo también es posible emplearlo en un computador aislados tipo Unix (e.g. OpenBSD, Linux).
- Pueden generarse boletines periódicos, finales o certificados con información configurable de acuerdo al colegio o en diversos formatos (e.g LaTeX, DocBook, HTML). Se incluye un ejemplo en LaTeX que puede adaptarse fácilmente.
- Permite especificar reglas para determinar valoraciones con base en otras (por ejemplo valoraciones de área a partir de valoraciones de indicadores). Las reglas pueden especificarse en un lenguaje propio (`prom`).
- Cuenta con una interfaz en PHP que facilita a los profesores el ingreso de logros, indicadores y valoraciones.
- Se ha diseñado para mantener privacidad, integridad y seguridad de la información.
- Los planes de estudio de cada materia podrían extraerse de contenidos que empleen las ayudas del programa `repasa` (ver [repasa]).
- Este proyecto ha sido desarrollado y es mantenido por voluntarios del proyecto Structio y la ayuda de otras personas y colegios. Se han empleado versiones de desarrollo en el colegio colombiano Gimnasio Fidel Cano desde el 2002 y en el tercer curso gratuito de administración de redes Linux en colegios.
- Las herramientas, fuentes, documentación y ejemplos han sido cedidos al dominio público por sus autores (ver Derechos y Créditos). Están disponible en: http://structio.sourceforge.net/sigue (http://structio.sourceforge.net/sigue)

1.1.2. Soporte y ayuda

Puede obtener soporte sin costo para usar `sigue` y eventualmente encontrará algún(a) voluntario(a) que quiera dar soporte personalizado en su colegio, en la lista <`structio-sopcol@lists.sourceforge.net`>. Puede suscribirse desde: http://lists.sourceforge.net/lists/listinfo/structio-sopcol Para mantenerse informado de otros desarrollos de Structio puede suscribirse a la lista de anuncios (tráfico bajo): http://lists.sourceforge.net/lists/listinfo/structio-anuncio (http://lists.sourceforge.net/lists/listinfo/structio-anuncio).

Dado que `sigue` es desarrollado por voluntarios y voluntarias, usted está invitado/a a dar su ayuda desinteresada. Para sugerir nuevas características puede emplear la categoría `sigue` del sistema de seguimiento de características (https://sourceforge.net/tracker/?atid=354503&group_id=4503&func=browse). Para reportar fallas puede emplear la categoría `sigue` del sistema para seguimiento de fallas (http://sourceforge.net/tracker/?func=browse&group_id=4503&atid=104503) o para fallas de seguridad el sistema de seguimiento de fallas de seguridad (http://sourceforge.net/tracker/?atid=615195&group_id=4503&func=browse). Si desea comunicarse con los desarrolladores escriba por correo electrónico a <`structio-info@lists.sourceforge.net`>

1.2. Sobre esta documentación

Esta documentación se ha escrito en DocBook empleando las herramientas y las convenciones de `repasa` (http://structio.sourceforge.net/repasa) para DocBook. En particular esta documentación tiene logros generales, indicadores de logro para cada capítulo y definiciones indexadas.

Logros

- Mantiene y chequea información de individuos, grupos, progreso en clasificaciones y plan de estudios empleando los formatos XML.
- Emplea la herramienta **iniperiodo** para preparar archivos de progreso.
- Genera reportes de acuerdo al uso que de a `sigue` bien con hojas XSLT o con la herramientas **reporte**.
- Emplea sigue para mantener progreso de los estudiantes de un colegio colombiano.

Capítulo 2. Registro del progreso de un curso

Público e indicadores de logro

Profesores y administrador de red

Indicadores de Logro

- Crea directorios y archivos con información de estudiantes.
- Crea archivos que mantienen grupos de estudiantes.
- Crea plan de estudios.
- Crea plantillas con progreso en clasificaciones y las actualiza.
- Emplea **iniperiodo** para generar conceptos en clasificaciones a partir de conceptos en ejercicios.
- Genera un resumen de resultados con hojas de estilo XSLT.

`sigue` puede emplearse para llevar el progreso de un curso. Los pasos típicos para esto, suponiendo que se emplean logros e indicadores son:

- Crear el archivo de clasificaciones con ejercicios asociados a los indicadores y el correspondiente archivo de secuencia.
- Preparar estructura de directorios, plan de estudios, archivos para estudiantes y archivo de grupo.
- Valorar ejercicios a medida que los estudiantes los desarrollan.
- Al final del curso (o de cada periodo), calcular concepto para cada Indicador con base en las valoraciones de los ejercicios.
- Generar un resumen de los resultados de todo el curso.

En este capítulo se presenta en detalle como realizar cada uno de estos pasos, tomando como ejemplo la información del tercer curso gratuito de Linux en colegios de Colombia (ver [CGLin3]) la cual está disponible en el directorio de ejemplos de `sigue`.

2.1. Creación de archivos de clasificación y secuencia

El primero de estos pasos puede hacerse creando directamente los archivos XML de `repasa` o extrayéndolos a partir de un documento en LaTeX o en DocBook. La forma de crearlos y mantenerlos se explica en detalle en la documentación de `repasa` (ver [repasa]).

Para este ejemplo supondremos que ya se tienen los siguientes archivos en un directorio `cglin3`:

- `cglin3.cla` con los logros, indicadores, ejercicios asociados a indicadores y eventualmente definiciones asociadas a cada indicador.
- `cglin3.sec` con el orden en el que se estudiarán los indicadores del archivo de clasificaciones separados en periodos.

2.2. Creación de información de estudiantes

Se requiere información de cada estudiante organizada en un directorio para el estudiante así como el listado del curso y el plan de estudios. La información que debe estar en el directorio de cada estudiante es: (1) datos del estudiante en un archivo para individuos y (2) progreso en el curso en un archivo de progreso en clasificaciones.

Por ejemplo dentro del directorio `cglin3` se ha creado el directorio `participantes` y dentro de este un subdirectorio para cada participante (usando el login como identificación). Cada directorio de participantes tiene un archivo `datos.ind` con los datos del estudiante y un archivo `cglin3.prc` con el progreso en las clasificaciones (pueden crearse automáticamente plantillas para estos archivos de progreso como se explica a continuación).

En el directorio `cglin3` se han ubicado el plan de estudios `cglin3.planest` y el listado del grupo `cglin3.grp`.

En las siguientes subsecciones se presentan detalles de los diversos archivos.

2.2.1. Datos de cada estudiante

El archivo de datos de cada estudiante puede incluir toda la información no académica que requiera. Es preferible que todos los archivos de estudiantes tengan los mismos tipos de información, los cuales se separan con el elemento `dato`. Un archivo típico del ejemplo considerado es:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE individuo SYSTEM "individuo.dtd">
<individuo mindatos="login;apellidos;nombres;pais;difusor;masinfo">
<dato tipo="correo">micorreo@micolegio.edu.co</dato>
<dato tipo="Homepage"></dato>
<dato tipo="Colegio"></dato>
<dato tipo="login">juaper</dato>
<dato tipo="nombres">Juan</dato>
<dato tipo="masinfo">si</dato>
<dato tipo="anterior"></dato>
<dato tipo="pais">Colombia</dato>
<dato tipo="apellidos">Perez</dato>
<dato tipo="practico">si</dato>
<dato tipo="difusor"></dato>
</individuo>
```

Hay un archivo como este en el directorio de cada estudiante con el nombre `datos.ind`.

Para verificar que un archivo para individuos (digamos `participantes/ESLAVA/datos.ind`) tenga sintaxis correcta emplee la herramienta **sigchq** así:

```
sigchq participantes/ESLAVA/datos.ind
```

Note que los tipos empleados en el archivo se pueden especificar en el atributo `mindatos` del elemento raíz, separando uno de otro con punto y coma. También podría validarse que haya al menos esos datos en el momento de chequear el archivo con **sigchq** y la opción `-datos`.

Es recomendable que el archivo con datos del estudiante incluya el dato `login` con una identificación única para el estudiante. Esta identificación puede ser empleada por programas como **iniperiodo** (para modifcar calificaciones de un sólo estudiante con la opción `-estudiante`).

Puede consultar más sobre este tipo de archivos en el manual de referencia individuo(5).

2.2.2. Datos del grupo

En una archivo para grupos se referencian todos los archivos con datos de estudiantes del curso. Parte del archivo de grupo del ejemplo considerado (`cglin3.grp`) es:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE grupo PUBLIC "-//Structio//DTD grupo 1.0" "grupo.dtd">
<grupo>
  <desc>Participantes del tercer curso gratuito de Linux</desc>

  <refind arch="participantes/CROACIA/datos.ind"/>
  <refind arch="participantes/ESLAVA/datos.ind"/>
  <refind arch="participantes/Fernando/datos.ind"/>
  <refind arch="participantes/ICCCOMPU/datos.ind"/>
  <refind arch="participantes/Marce/datos.ind"/>
  <refind arch="participantes/ORLANDO/datos.ind"/>
  <refind arch="participantes/Ocalex/datos.ind"/>
</grupo>
```

Para revisar la sintaxis de un archivo para grupos emplee:

```
sigchq cglin3.grp
```

Note que antes de revisar un archivo para grupo debe haber creado los archivos con datos de estudiantes.

Puede consultar más sobre este tipo de archivos en el manual de referencia grupo(5).

2.2.3. Plan de estudios

Para relacionar asignaturas, con cursos, con grupos de estudiantes y con profesores se emplea un plan de estudios. En el caso de un sólo curso este archivo (`cglin3.planest`) es muy sencillo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE planest PUBLIC "-//Structio//DTD plan de estudios 1.1" "planest.dtd">
<planest tipos_curso="Nivel;Grado" tipos_asignatura="Área;Asignatura">

  <desc>Plan de estudios del tercer curso gratuito de administración de redes Linux en coleg
```

```
  <biblio>Aprendiendo a aprender Linux. http://structio.sourceforge.net/AA_Linux_colegio</bi
  <derechos tipo="Dominio público" tiempo="2002">Cedido al dominio público</derechos>
  <autor fecha="2002">Varios voluntarios. </autor>

  <curso tipo="CursoLinux" id="cglin3" refgrupo="cglin3.grp">
     <desc>Tercer curso gratuito de Linux</desc>
     <asignatura tipo="Tema" id="Linux" refsectemas="cglin3.sec"
             intensidad="3" prc="cglin3.prc">
             <desc>Tercer curso gratuito de Linux</desc>
      </asignatura>
  </curso>
</planest>
```

Note que el archivo del grupo (`cglin3.grp`) se referencia en el atributo `refgrupo` del elemento `curso`. El archivo de secuencia `cglin3.sec` se referencia como atributo `refsectemas` de una asignatura.

Puede consultar más sobre este tipo de archivos en el manual de referencia planest(5).

2.2.4. Archivos de progreso en clasificaciones

Cada estudiante debe tener un archivo de progreso en las clasificaciones del curso. Estos archivos deben ser creados al comienzo del curso y se actualizan a medida que el curso transcurre. Parte de un archivo de progreso se presenta a continuación (la extensión sugerida es `.prc`):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE prcla PUBLIC "-//Structio//DTD progreso en clasificaciones 1.0" "prcla.dtd">

<prcla conceptos="Logro:(g='E' || g='B' || g='I');Indicador:(g='E' || g='B' || g='I');ejerci
  <progcla idcla="cglin3.cla:recursos" tipo="Logro">
    <progcla idcla="cglin3.cla:recursos.plan" tipo="Indicador">
      <progejer concepto="E" fecha="25/4/2002" idejer="cglin3.cla:recursos.plan.doc"></proge
      <val concepto="E" periodo="Preparación">
        <anota valor="Preparación" tipo="prom. ej."></anota></val></progcla>
    <progcla idcla="cglin3.cla:recursos.escribir-leer" tipo="Indicador">
      <progejer concepto="E" fecha="25/4/2002" idejer="cglin3.cla:recursos.escribir-leer.vis
      <progejer concepto=" " fecha="" idejer="cglin3.cla:recursos.escribir-leer.enlaces"></p
    </progcla>
  </progcla>
  <progcla idcla="cglin3.cla:usalinux" tipo="Logro">
    <progcla idcla="cglin3.cla:usalinux.doc" tipo="Indicador">
      <progejer concepto="E" fecha="15/5/2002" idejer="cglin3.cla:usalinux.doc.guías"></prog
     <progejer concepto=" " fecha="" idejer="cglin3.cla:usalinux.doc.ayuda"></progejer>
      <val concepto="B" periodo="Básico">
<anota valor="Básico" tipo="prom. ej."></anota></val>
    </progcla>
  </progcla>
</prcla>
```

Tenga en cuenta que los archivos de progreso deben ser actualizados a medida que el curso transcurre.

Puede consultar más sobre este tipo de archivos en el manual de referencia prcla(5).

2.2.4.1. Generación automática de plantillas para archivos de progreso

Pueden crearse plantillas de los archivos de progreso con espacio para completar las valoraciones. Esto puede hacerse empleando una hoja de estilo XSLT como: `xslt/cla2prc.xsl` (ubicada en el directorio en el que se instalaron los datos de sigue, e.g `/usr/local/share/sigue`). Esta hoja podría usarse con **xsltproc** para generar el archivo de progreso del estudiante con datos en `participantes/ESLAVA` así:

```
export SGML_CATALOG_FILES="/usr/local/share/xml/structio/catalog"
xsltproc -catalogs -stringparam filename cglin3.cla \\
/usr/local/share/sigue/xslt/cla2prc.xsl \\
cglin3.cla > participantes/ESLAVA/cglin3.prc
```

la variable `SGML_CATALOG_FILES` debe contener la ruta al catálogo de DTDs de Structio. De emplear el parámetro `filename` este debe tener como valor el nombre del archivo de clasificaciones (en este caso `cglin3.cla`) que se agregará a la identificación de cada clasificación y ejercicio referenciado.

Pueden crearse todos los archivos de progreso después de haber creado la estructura de directorios con un script para el interprete de comandos, por ejemplo:

```
#!/bin/sh
export SGML_CATALOG_FILES="/usr/local/share/xml/structio/catalog"
for i in participantes/*; do
  if (test -f $i/datos.ind) then {
    echo $i;
    xsltproc -catalogs -stringparam filename cglin3.cla \\
    /usr/local/share/sigue/xslt/cla2prc.xsl \\
    cglin3.cla > participantes/$i/cglin3.prc
    } fi;
done;
```

2.3. Valoraciones en indicadores

`sigue` permite computar automáticamente las valoraciones en indicadores de un periodo a partir de valoraciones de los ejercicios asignados para ese periodo (como se explica en la siguiente sección). También es posible agregar manualmente valoraciones de indicadores en los archivos de progreso. De hecho en algunos casos (recuperaciones o revaloración de indicadores) ese es el único mecanismo disponible.

Al modificar manualmente un archivo de progreso, tenga en cuenta:

- Por cada indicador del periodo que se valora, agregue sólo una valoración (empleando el semestre que se valora en el atributo `periodo`).

- No modifique valoraciones de periodos ya cerrados. En caso de requerir revalorar un indicador, agregue una valoración en el periodo en el que se hace la re-valoración y deje intactas las valoraciones de periodos anteriores.
- Después de hacer cambios, verifique que el archivo de progreso tenga sintaxis correcta con **sigchq**.

2.3.1. Registro de progreso en ejercicios y cálculo automático de valoraciones de indicadores

Al automatizar la valoración de indicadores, supusimos que mientras el curso se desarrolla los estudiantes realizan ejercicios asignados de acuerdo al tema que se estudie. Estos ejercicios son revisados por el profesor (o puede ser revisados automáticamente) y valorados en los archivos de progreso. La valoración de cada ejercicio se registra en el archivo de progreso de cada estudiante junto con la fecha usando el elemento `progejer` y teniendo en cuenta las recomendaciones para modificar manualmente archivos de progreso, por ejemplo dentro del elemento `progcla` para el indicador `usalinux.doc` podría agregarse una valoración para el ejercicio `usalinux.doc.guías`:

```
<progejer idejer="usalinux.doc.guías" concepto="E" fecha="15/5/2002"/>
```

Cada vez que se termina un periodo del curso, se calculan valoraciones de los indicadores a partir de las valoraciones en los ejercicios. Para esto se prepara un script para calcular promedios (escrito en lenguaje `prom`) y se usa el programa **iniperiodo**. Por ejemplo el programa podría estar en un archivo `cglin3.prom` y ser:

```
/* Calcula 'promedios' en indicadores a partir de valoraciones de ejercicios.
*/

aserción(ejecutor=="iniperiodo");

tot=0.0;
nume=0;
numb=0;
i=0;
mientras (i<tamaño(ejer)) {
si (ejer[i]=="E") {
nume=nume+1;
}
sino {
si (ejer[i]=="B") {
numb=numb+1;
}
}
i=i+1;
}
tot=ejer_en_cla;
```

```
pe=nume/tot;
pb=numb/tot;
si (pe>=0.8) {
retorna "E";
}
si (pb+pe>=0.6) {
retorna "B";
}
retorna "I";
```

Para probar el script sin escribir en los archivos de progreso, computando valoración de indicadores para el periodo ‘Preparación´ del estudiante con login ‘CROACIA´ se podría emplear:

```
iniperiodo -noescribir -ejercicios -tipoval Indicador -prom cglin3.prom \
  -cron cglin3.sec -D ../../dtds/ -p Preparación -estudiante CROACIA \
  -nopendientes -val " " cglin3.planest
```

Note que se emplea un cronograma (`cglin3.sec`) el cual debe definir la fecha de inicio y finalización de cada periodo (pues en el promedio sólo se tienen en cuenta los ejercicios realizados durante el periodo, y en el archivo de progreso el progreso en ejercicios se anota con la fecha de realización). Puede ver algunas ayudas para depurar programas prom en Sección 3.5.2.1.

Una vez se verifique que el calculo de promedios es apropiado y que el script funciona correctamente pueden calcularse las valoraciones y agregarse a los archivos de progreso de todos los estudiantes del curso con:

```
iniperiodo -ejercicios -tipoval Indicador -prom cglin3.prom \
-cron cglin3.sec -D ../../dtds/ -p Preparación -nopendientes
-val " "  cglin3.planest
```

> **Aviso**
>
> A menos que se use la opción `-noescribir`, **iniperiodo** modificará los archivos de progreso. Y a menos que se especifique una asignatura particular (con la opción `-asignatura`) o un estudiante particular (con la opción `-estudiante`), **iniperiodo** modificará los archivos de progreso de todos los estudiantes en todas las asignaturas.

2.4. Generación de un resumen de resultados

Una vez estén en los archivos de progreso las valoraciones de indicadores o a medida que se llenen, es posible generar un reporte en HTML bien con el programa **reporte**, o por ejemplo empleando un script y una hoja de estilo XSLT como se presenta en esta sección.

Por ejemplo para generar un resumen con la valoración de los indicadores de un estudiante puede emplearse una hoja de estilo XSTL como la disponible en `xslt/indicador2table.xsl`:

```
xsltproc -catalogs /usr/local/share/sigue/xslt/indicador2table.xsl participantes/ESLAVA/cgli
```

Esta hoja de estilo generará una fila para una tabla HTML con las valoraciones de los indicadores. Recuerde que antes requiere establecer la variable `SGML_FILE_CATALOGS` como se presenta en Sección 2.2.4.1.

Para generar una tabla completa con los resultados de todos los estudiantes puede emplearse un script como el siguiente (disponible en `ejemplos/cglin3/resumen.sh`):

```
#!/bin/sh
# Genera un resumen con valoraciones de indicadores
# Dominio público. 2003. http://structio.sourceforge.net/sigue Sin garantías

export SGML_CATALOG_FILES=$SGML_CATALOG_FILES:../../dtds/catalog
echo "<html><body>"
echo "<table border='1'><tr><td>Estudiante</td>
<td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><t

for i in participantes/*; do
if (test -f $i/datos.ind) then {
b=`basename $i`;
echo "
<tr><td>$b</td>";
xsltproc -catalogs herram/indicador2table.xsl $i/cglin3.prc
if (test "$?" != "0") then {
echo "Problemas $i" > /dev/stderr
exit 1;
} fi;
} fi;
done
echo "</table></body></html>";
```

Note que primero se genera el encabezado del archivo HTML y de la tabla, después se generan los resultados de cada estudiante (una fila por estudiante) y finalmente se cierra la tabla y el archivo HTML. Para cada estudiante se genera el nombre del directorio y empleando la hoja de estilo se genera la fila de valoraciones en los indicadores.

Para usarlo podría ejecutarse:

```
./resumen.sh > resumen.html
```

que generaría el resumen en el archivo `resumen.html`, el cual podría examinarse con un navegador.

Capítulo 3. Registro de progreso en un colegio colombiano

Público e indicadores de logro

Administrador de red

Indicadores de Logro

- Planea la utilización de `sigue` en un colegio orientado por la legislación colombiana vigente.
- Prepara cronograma e información de estudiantes y profesores.
- Mantiene plan de estudios.
- Genera reportes de periodos o de fin de año.

3.1. Contexto

En este capítulo se presenta como puede emplearse `sigue` para registrar el progreso de los estudiantes de un colegio colombiano.

De acuerdo a la ley 115 de 1994 (ver [[L115-1994]]), artículo 11 (http://structio.sourceforge.net/leg/1994ley115.html#A11) los niveles de la educación formal son:

1. El preescolar que comprende mínimo un grado obligatorio.
2. La educación básica con una duración de nueve (9) grados que se desarrollará en dos ciclos: La educación básica primaria de cinco (5) grados y la educación básica secundaria de cuatro (4) grados.
3. La educación media con una duración de dos (2) grados.

De acuerdo al artículo 23 (http://structio.sourceforge.net/leg/1994ley115.html#A23) de la misma ley las áreas obligatorias y fundamentales (que sean más del 80% del plan de estudios) son:

- Ciencias naturales y educación ambiental.
- Ciencias sociales, historia, geografía, constitución política y democracia.
- Educación artística.
- Educación ética y en valores humanos.
- Educación física, recreación y deportes.
- Educación religiosa (no se obliga).
- Humanidades, lengua castellana e idiomas extranjeros.
- Matemáticas.
- Tecnología e informática.

Además cada institución podría agregar otras áreas de acuerdo al PEI particular, y podrá especificar en el mismo PEI asignaturas que desarrollen cada área en cada grado (ver artículo 76 (http://structio.sourceforge.net/leg/1994ley115.html#A76) de la misma ley). Estas asignaturas deberán desarrollar un plan de estudios que debe contar con logros acordes con indicadores de logro (artículo 38 (http://structio.sourceforge.net/leg/1994dec1860.html#A38) del decreto 1860 de 1994 – ver [D1860-1994]).

Con respecto a la evaluación el capítulo VI (http://structio.sourceforge.net/leg/1994dec1860.html#A38) del decreto 1860 de 1994 indica que debe ser continua, integral y cualitativa, que periódicamente deben generarse informes legibles para padres y deben valorarse los indicadores de logro teniendo en cuenta los diferentes ritmos de aprendizaje y permitiendo a cada estudiante recuperar en un periodo lo que no haya quedado bien en periodos anteriores o incluso ser promovido anticipadamente a otro grado. De acuerdo al artículo 51 del mencionado decreto, los informes de final de año deben valorar logros de cada asignatura con la escala insuficiente, bien y excelente. La promoción de un grado a otro es evaluada por una comisión de promoción.

Aunque en nuestra opinión los métodos de evaluación propuestos por el decreto 1860 de 1994, pueden ser apropiados, la resolución 230 de 2002 desmontó el requerimiento de logros e indicadores de logros, modificó la escala de valoración indicando que se realizara por áreas (excelente, sobresaliente, aceptable, insuficiente, deficiente) e impone un mínimo de promoción del 95% de los educandos en cada grado.

`sigue` puede emplearse para registrar progreso de estudiantes de acuerdo a cualquiera de los dos modos de evaluación (tanto la del decreto 1860 de 1994 como la de la resolución 230 de 2002) así como otros que puedan haberse acordado en el PEI de instituciones particulares. En este capítulo presentamos y sugerimos una mezcla de ambos modos, que por un lado consideramos permite una evaluación continua basada en los indicadores de logros, adaptada al ritmo de cada estudiante (permitiendo acumular indicadores pendientes de un periodo o grado a otro) y que empleando "reglas" permite satisfacer los requerimientos automatizables de la resolución 230 de 2002.

3.2. Planeando el uso de sigue

Para emplear sigue primero deben acordarse algunas convenciones particulares para su colegio:

- De no haberlo hecho acordar modo y escala (o escalas) de valoración. Sugerimos que los profesores valoren y revaloren continuamente indicadores de logro con la escala insuficiente, bueno, excelente. En el boletín periódico además de la valoración de cada indicador, podrá computarse una valoración por área con la escala excelente, sobresaliente, aceptable, insuficiente y deficiente. Tal computo puede hacerse para cada área con reglas basadas en las valoraciones de los indicadores de las diversas asignaturas del área. Tales reglas pueden ser acordadas por la comunidad educativa.
- Decidir que información desea mantener para presentar en los registros valorativos. Sugerimos mantener valoraciones de indicadores de logro de cada estudiante periodo a periodo, así como registro de inasistencia a clase y observaciones. Manteniendo esta información es posible generar cada periodo un reporte como el presentado a continuación:

- Acordar un proceso para mantener la información así como responsables. Sugerimos que un administrador con experiencia en sistemas tipo Unix instale y mantenga `sigue`, periódicamente puede verificar y sacar copia de la información así como mejorar seguridad, integridad y privacidad. El mismo administrador podría mantener el cronograma, cuadrar detalles de presentación de los boletines e imprimirlos al final de cada periodo o del año. Sugerimos que los profesores modifiquen logros e indicadores antes de comenzar cada periodo y que ellos mismos valoren o reevaluen indicadores de los estudiantes a medida que transcurre el periodo.
- Acordar la identificación que se empleará para cada periodo de cada año (debe ser una identificación XML válida). Se sugiere una como p*año-periodo* (por ejemplo `p2007-1` para identificar primer periodo del año 2007), y una división de cada año en 4 periodos. Esta identificación será usada en diversos archivos (en particular en el cronograma) y puede usarse para marcar el repositorio de CVS.
- Acordar la identificación que se empleará para los diversos grados (debe ser una identificación XML válida), se sugiere `g1` para primero de primaria, `g2` para segundo y así sucesivamente. Si hay diversos cursos en algunos grados, podrían ser `g3a` y `g3b`.
- Acordar la forma como se identificaran los estudiantes de manera única (teniendo en cuenta que de un año a otro cambiarán y que puede haber homónimos). Puede ser bien con un número (por ejemplo de carnet) o por ejemplo si todos los estudiantes tienen accesos a computadores del colegio podría ser el año de nacimiento, el login y eventualmente un dígito para diferenciar repetidos.
- Acordar la forma como se identificarán los profesores. Puede ser una identificación análoga a la de estudiantes.

3.3. Inicialización

Para crear una estructura de directorios se emplea el script **sigue-colegio.sh**, después se modifica y completa la información generada y opcionalmente se crea un módulo en un repositorio CVS. A continuación se describen los pasos, suponiendo que el directorio con la información del colegio (i.e el *directorio base*) será `/var/nuestra-escuela` y se comenzará a emplear `sigue` en el año 2007.

1. Cree la estructura de directorios desde la cuenta `root`así:

   ```
   # cd /var
   # sigue-colegio.sh nuestra-escuela 2007
   ```

   Esto creará el directorio `/var/nuestra-escuela` con información de ejemplo para comenzar a registrar progreso en el año 2007.

> **Aviso**
>
> Si cuenta con un servidor web, sugerimos que emplee una ruta que no pueda accederse desde un navegador. En especial si planea usar la interfaz PHP (ver Sección 3.6).

2. Edite el cronograma `/var/nuestra-escuela/cron.sec` especificando la división en periodos del año, así como el inicio y fin de cada periodo (con los atributos `inicio` y `fin` del elemento `sec`). Puede verificar el archivo con la herramienta **repchq**.

3. En la ruta `/var/nuestra-escuela`, dentro del subdirectorio `estudiante`, cree un directorio por cada estudiante. Para evitar duplicados, emplee la identificación de cada estudiante en el nombre del directorio. En el directorio de cada estudiante cree el archivo `datos.ind` (ver Sección 2.2.1), al menos con los datos: `Apellidos`, `Nombres`, `login` y `FechaNacimiento`. Las plantillas generadas emplean como identificación el año de nacimiento y el login del estudiante, puede guiarse en estas y después borrarlas o puede automatizar este proceso como se explica más adelante.

   Emplee la herramienta **sigchq** para verificar el formato de los archivos que modifique.

4. En los subdirectorios `grupo/2007/bprimaria`, `grupo/2007/bsecundaria` y `grupo/2007/media` cree un archivo por cada grado, cada archivo debe referenciar los estudiantes del grado respectivo (se sugieren que en los nombres se use la identificación de cada grado/curso, por ejemplo `g1.grp`, `g5a.grp`, etc). Al crear estos archivos y en general al referenciar un archivo desde otro tenga en cuenta emplear una ruta relativa al directorio `planest/2007`[1].

   Emplee la herramienta **sigchq** para verificar los grados que edite.

5. Para completar la información de profesores proceda de forma análoga al caso de estudiantes, dentro del subdirectorio `profesor`, cree un directorio para cada profesor y el archivo de datos `datos.ind` al menos con los mismos datos del archivo de datos de estudiantes. Modifique también el archivo `grupo/2007/profesores.grp` para que referencia a todos los profesores del colegio.

6. Adapte los subdirectorios y datos del directorio `planest/2007` para que correspondan a los niveles, grados y asignaturas de la institución. Para comenzar no es indispensable modificar los archivos de asignaturas (e.g `castellano.cla` y `castellano.sec`) sólo, de requerirlo, copie las plantillas disponibles para las asignaturas que hagan falta. Por ejemplo si en primero de primaria falta una asignatura con identificación `servicio`:

   ```
   cd planest/bprimaria/1
   cp castellano.cla servicio.cla
   cp castellano.sec servicio.sec
   ```

   Para verificar los archivos de clasificaciones y secuencia emplee la herramienta **repchq** y revise instrucciones en el manual de `repasa` (ver [[repasa]]).

7. Modifique el modelo de plan de estudios[2] disponible en `planest/2007/col.planest` (dividido en los archivos `planest/2007/g1.planest`, `planest/2007/g2.planest` y así sucesivamente hasta `planest/2007/g11.planest`). Este archivo debe referenciar los niveles, los grados (y cursos si los hay), las asignaturas, profesores y los archivos con logros e indicadores de las asignaturas de cada grado/curso.

1. Todos los scripts se han implementado suponiendo que las rutas son relativas a ese directorio. Si se desea podría asumirse una convención diferente y emplear la opción `-D` soportada por las herramientas de `sigue` (que permite especificar uno o más directorios en los cuales abrir archivos cuya ruta sea relativa).
2. El modelo de plan de estudios sigue las convenciones de niveles, grados y áreas obligatorias de la Ley 115 de 1994.

Después de editarlo verifíquelo con la herramienta **sigchq**.

8. Si en el computador en el que está la información tienen cuenta los profesores y quien administra `sigue` (que debe tener acceso a la cuenta root), es recomendable establecer permisos de los archivos que los profesores podrán editar. Vea la sección Sección 3.3.2.
9. Opcional: inicie un módulo en un repositorio CVS para (1) respaldar la información, (2) mantener información histórica más completa[3] y (3) facilitar trabajo en grupo (por ejemplo cada profesor podría trabajar su propia copia del plan de estudios). Puede basarse en el ejemplo presentado en la sección "Consideraciones sobre el control de versiones" del manual de `repasa` (ver [[repasa]]). Sugerimos tener una sola copia local (digamos en `/var/nuestra-escuela`) que sea editada por todos los profesores empleando permisos. Después de ediciones es recomendable actualizar el repositorio y cada vez que se cierre un periodo es recomendable agregar una marca a todo el repositorio.

3.3.1. Inicialización automática de datos de estudiantes y profesores

3.3.1.1. A partir de usuarios y grupos

Si todos los estudiantes tienen login en un computador o red tipo Unix y todos pertenecen a un grupo (digamos `estudiante`), la inicialización de datos para estudiantes/profesores puede automatizarse extrayendo los datos del archivo `/etc/passwd`. Si los profesores y estudiantes tienen cuentas pero aún no ha creado grupos se sugiere que lo haga, por ejemplo:

```
groupadd -g 1102 profesor
groupadd -g 1101 estudiante
```

Para eso se sugiere que en el campo GECOS de cada estudiante se emplee como nombre (i.e antes de la primera coma): *`nombres - apellidos - grado - fecha-nacimiento`* Información que puede ser modificada posteriormente con el programa `chfn` (el cual puede adaptarse para que lea y organice la información siguiendo la convención propuesta).

Para crear directorios y archivos `datos.ind` a partir de la información en `/etc/passwd` puede emplear el script `passwd2est.awk` el cual se ejecuta desde el directorio creado por **sigue-colegio.sh** así:

```
cd /var/nuestra-escuela
GID=1101 ANIO=2007 awk -f /usr/local/share/sigue/herram/passwd2est.awk /etc/passwd
```

siendo 1101 la identificación del grupo `estudiante`. Este script extraerá la información de usuarios del grupo 1101 y creará un directorio por cada año de nacimiento y dentro de cada uno de estos directorios para cada estudiante que haya nacido ese año con el login como nombre y con el archivos `datos.ind`. Así mismo este script creará los archivos de los grados en el directorio `grupo`, en el subdirectorio del año 2007.

3. Si no usa CVS empleando las convenciones que sugerimos sólo podrá mantener archivo histórico del estado del colegio al final de cada año (por ejemplo para imprimir reportes finales de años anteriores). Si emplea CVS y una política de actualización cada periodo, mantendrá archivo histórico del estado del colegio al final de cada periodo (por ejemplo para imprimir reportes de periodos de años anteriores).

> **Aviso**
>
> **passwd2est.awk** borrará información ya existente. En particular si existía un archivo para cada grado, la información será remplazada por la que este script extraiga.

De forma análoga puede extraerse la información de profesores con el script **passwd2prof.awk**, por ejemplo si el grupo de profesores tuviera identificación 1102:

```
cd /var/nuestra-escuela
GID=1102 awk -f /usr/local/share/sigue/herram/passwd2prof.awk /etc/passwd
```

El campo GECOS de profesores debe tener la misma sintaxis del sugerido para estudiantes (el grado no será tenido en cuenta).

También es posible a partir de un archivo para grupos que referencie archivos de individuo con Nombres, Apellidos, FechaNacimiento y login, generar una secuencia de comandos que cree las cuentas. Para esto puede emplear o adaptar la hoja de estilo `xslt/grp2debuser`, la cual genera comandos con la sintaxis requerida en sistemas Linux/Debian y recibe como parámetros el grupo al que perteneceran los usuarios y opcionalmente el grado:

```
cd grupo/2002
xsltproc -catalogs -stringparam grupo profesor \\
  /usr/local/share/sigue/xslt/grp2debuser.xsl profesores.grp
```

3.3.1.2. A partir de un listado en formato CSV

Si no ha creado cuentas para profesores ni estudiantes, puede generar los archivos para sigue y scripts para crear cuentas usando `herram/lista2est.awk` y `herram/lista2prof.awk`

3.3.2. Permisos

Sugerimos que cada profesor tenga una cuenta en el computador donde está la información y que exista un grupo `profesor` al cual pertenezcan sólo los profesores. De ser así puede generar el archivo `perm.sh` que establece permisos y propietarios para limitar posibilidad de ver y editar archivos con plan de estudios y archivos de progreso, siguiendo algunas recomendaciones que al respecto se presentan en el manual de `repasa` (ver [repasa]).

El script `planest/2007/perm.sh` se genera con una hoja de estilo XSLT, a partir del plan de estudios `planest/2007/col.planest`. Genérelo cada vez que modifique el plan de estudios con:

```
cd planest/2007/
export SGML_CATALOG_FILES=/usr/local/share/xml/structio/catalog
xsltproc -catalogs /usr/local/share/sigue/xslt/planest2perm.xsl col.planest > perm.sh
```

o bien con el script `planest/creaperm.sh`:

```
cd planest/2007
../creaperm.sh
```

El script generado `perm.sh` establecerá propietarios y permisos de los archivos del plan de estudios referenciados en `col.planest`, así como los archivos de progreso de estudiantes que estén en grados referenciados en `col.planest`, dando permiso de escritura a cada profesor sólo en los archivos del plan de estudio que maneja y sólo en los archivos de progreso de los estudiantes que orienta. El script `perm.sh` debe ser ejecutado desde la cuenta `root`. Este script creará archivos de secuencia y clasificación del plan de estudio que puedan hacer falta. Si el plan de estudio tiene definido un login de quien lo maneje, este script establecerá permisos para manejar el plan de estudio solo a ese usuario.

Una vez haya generado `planest/2007/perm.sh` puede modificar propietarios y permisos de los diversos archivos y directorios así:

```
cd /var/nuestra-escuela
chgrp -R profesor .
chmod -R o-rwx .
find . -type d -exec chmod gu+x {} ";"
cd planest/2007
. ./perm.sh
```

La primera vez que ejecute **perm.sh** aparecerán varios errores porque los archivos de progreso no se han creado. Estos archivos serán creados cuando ejecute por primera vez **iniperiodo**, como se explica en Sección 3.4.3

3.3.3. Personalización

Es necesario personalizar `sigue` de acuerdo a los planes que realizó (ver Sección 3.2) en el archivo `/var/nuestra-escuela/planest/param.sh` editelo y configure:

- Listado de valoraciones que indican indicador pendiente (o perdido). En la variable `pendientes` separando una de otra con punto y coma (;).
- Valoración que se pondrá por defecto (antes de que el profesor ingrese valoraciones). En la variable `valdef`

3.4. Administración de la información del colegio

3.4.1. Administración de estudiantes y profesores

Cuando un nuevo estudiante o profesor ingrese a la institución debe crearse un directorio bien en `estudiante` o en `profesor` y debe referenciarse la nueva información en el archivo del grado o del grupo de profesores y en el caso de profesores también en el plan de estudios.

Cuando salga un estudiante o un profesor, no debe borrarse su directorio, porque hace parte de la memoria de la institución y puede requerirse en el futuro. Lo que debe hacerse es retirarlo del archivo del grupo y/o del plan de estudios (que de usarse CVS debería ser actualizado en el repositorio al final de cada periodo).

Puede generar listados en HTML de los estudiantes de cada grado y del grupo de profesores, digamos en el directorio `/var/www/ssl/sigue/listas` con:

```
# cd /var/nuestra-escuela/planest/2007/
# ../../grupo/genlistas.sh /var/www/ssl/sigue/listas/ 2007
```

3.4.2. Administración del plan de estudios

El plan de estudios en `sigue` está dividido en: (1) un archivo que referencia grados y las asignaturas por grados y (2) archivos de cada asignatura con logros e indicadores.

El archivo que referencia grados y asignaturas[4] por grados (`planest/2007/col.planest`) normalmente se fija al comienzo del año aunque en algunos casos puede cambiarse de un periodo a otro.

Los archivos de cada asignatura con logros e indicadores normalmente se actualizan cada periodo (para registrar los logros e indicadores del periodo siguiente), labor que pueden realizar los profesores. Por ejemplo el profesor de castellano de primero podría modificar cada periodo los archivos `planest/2007/bprimaria/1/castellano.cla` y `planest/2007/bprimaria/1/castellano.sec` para actualizar los logros e indicadores del periodo.

Los indicadores que no sean recuperables deben marcarse con una anotación (el siguiente ejemplo es tomado del archivo `bprimaria/1/comportamiento.cla` del plan de estudios de la plantilla generada por **sigue-colegio.sh**):

```
<clasif tipo="Indicador" id="Comportamiento.Disciplina">
<desc>Disciplina</desc>
  <anota tipo="Recuperable" valor="no"/>
</clasif>
```

para que la herramienta **iniperiodo** (que se presenta en la siguiente sección) los identifique.

Si en alguna asignatura, se empleen uno o más indicadores no recuperables en más de un periodo (como podría ser el caso de disciplina), puede referenciar el mismo varias veces en el archivo de secuencia (ver

4. Además de asignaturas pueden referenciarse otros elementos que deben evaluarse o aparecer en el registro valorativo, por ejemplo comportamiento. Estas pueden referenciarse con el elemento `asignatura` pero empleando un tipo diferente, por ejemplo `evaluable`. Un ejemplo puede verse en el archivo con el plan de estudios de la plantilla generada por **sigue-colegio.sh**.

por ejemplo archivo `bprimaria/1/comportamiento.sec` del plan de estudio de la plantilla generada por **sigue-colegio.sh**).

Los archivos que se modifiquen deben ser verificados con la herramienta **sigchq**, eventualmente con `Makefiles` o scripts como se presenta en en la sección sobre planes de estudio del manual de `repasa` ([repasa]).

Una vez se han llenado los planes de estudio con logros e indicadores (sugerimos emplear la interfaz PHP), es buena práctica imprimirlos para revisión por parte de coordinadores y/o profesores. Puede generar listados en HTML de los logros e indicadores de cada grado, por ejemplo en el directorio `/var/www/ssl/sigue/indicadores/` con:

```
# cd /var/nuestra-escuela/planest/2007/
# ../genplan.sh /var/www/ssl/sigue/indicadores/ 2007
```

o para imprimir indicadores sólo del 3er periodo:

```
# ../genplan.sh /var/www/ssl/sigue/indicadores/ 2007 3
```

El listado de asignaturas que no tengan desempeños en un periodo (digamos 3) podría sacarse con:

```
cd /var/nuestra-escuela/planest/2007/
for i in ‘find . -name "*sec"‘; do
  if (test "‘grep -n "p2007-3" $i‘" = "") then { echo $i; } fi;
done
```

Si cuenta con el programa **ispell**, puede revisar ortografía de logros e indicadores con:

```
cd /var/nuestra-escuela/planest/2007
../revisaort.sh
```

El diccionario del colegio queda en `/var/nuestra-escuela/dic.ispell` a este puede agregar palabras que requiera, una palabra por línea. Las palabras con tilde se agregan de forma especial, por ejemplo la palabra línea se agrega como:

```
l’inea
```

3.4.3. Administración de valoraciones

Durante cada periodo se deben valorar indicadores de cada asignatura y de cada estudiante. Esto se hace modificando los archivos de progreso de los estudiantes para:

1. Agregar referencias a indicadores del periodo y sus valoraciones.
2. Valorar de nuevo indicadores pendientes de periodos anteriores. La idea es no modificar la información existente sino agregar una nueva valoración para el periodo que se trabaja.

3. Empleando anotaciones agregar otra información que haya acordado mantener al planear el uso de sigue (ver Sección 3.2). Por ejemplo para mantener registro de fallas sugerimos que agregue en la raíz de cada archivo de progreso de cada estudiante y cada asignatura una anotación como esta por cada periodo (que especifica 2 fallas en el primer periodo de 2007):

   ```
   <anota tipo="Fallas" valor="p2007-1">2</anota>
   ```

   De manera análoga pueden agregarse observaciones por asignatura con

   ```
   <anota tipo="Observacion" id="p2007-1">Hemos aprendido con él/ella</anota>
   ```

Sugerimos emplear la interfaz PHP para editar los archivos de progreso. Para editarlos manualmente puede ver recomendaciones en Sección 2.3. Tenga en cuenta que antes de editar manualmente los archivos, puede automatizarse parte de la labor con el programa **iniperiodo**, que agregará referencias a indicadores del periodo y plantillas para completar valoraciones de los indicadores pendientes y del periodo. Esta herramienta debe emplearse después de que todos los profesores hayan ingresado la versión final de los logros e indicadores del periodo. Por ejemplo para preparar los archivos de progreso de toda la información referenciada en el plan de estudios (todos los estudiantes y todas las asignaturas) para el primer periodo del 2007[5]:

```
cd /var/nuestra-escuela/planest/2007
iniperiodo -anota "Fallas:0;Observaciones:" -pendiente "I" -norec "Recuperable;no" -tipoval
```

o para mayor brevedad emplee el script **inip.sh** para inicializar el periodo `p2007-1` así:

```
cd /var/nuestra-escuela/planest/2007
../inip.sh p2007-1
```

Por ejemplo para limitar los archivos que se preparan a los del grado 10 en la asignatura `democracia`:

```
iniperiodo -anota "Fallas:0;Observaciones:" -asignatura democracia -curso g10 -pendiente "I'
```

Después de ejecutar **iniperiodo** es recomendable que establezca permisos (ver Sección 3.3.2).

3.4.3.1. Revisión de calificaciones

Una vez los profesores ingresen valoraciones es recomendable generar un listado de las calificaciones generadas y solicitarles firmarlo. Este listado puede generarse con:

```
cd /var/nuestra-escuela/planest/2007/
mkdir /var/www/ssl/sigue/valoraciones/10may2007
```

5. La orden presentad para preparar todos los archivos de progreso está en el script `planest/2007/inip.sh` el cual sólo recibe el periodo por preparar.

```
../gennotas.sh /var/www/ssl/sigue/valoraciones/10may2007 2007 1
```

También es posible generar un archivo con las frecuencias de valoración en cada indicador, así como los estudiantes que no aprobaron algún indicador con:

```
cd /var/nuestra-escuela/planest/2007/
../gennotas.sh /var/www/ssl/sigue/estad 2007 1
```

3.4.4. Inicio de un nuevo año

La forma más simple de inicializar la información para un nuevo año es copiando parte de la información del año anterior. No es necesario replicar información de estudiantes ni de profesores, sólo los grupos, el plan de estudios y el directorio de reportes.

Los periodos del nuevo año se agregan al cronograma `/var/nuestra-escuela/cron.sec`, sin eliminar los periodos de años anteriores.

3.4.5. Verificación de la información

Es posible verificar la información mantenida con `sigue` usando los archivos `Makefile` disponibles en diversos directorios. Estos se usan con el programa **make**[6]:

- Para verificar toda la información:

  ```
  cd /var/nuestra-escuela
  make
  ```

- Para verificar datos de estudiantes:

  ```
  cd /var/nuestra-escuela/estudiante
  make valida-individuo
  ```

- Para verificar datos de profesores:

  ```
  cd /var/nuestra-escuela/profesor
  make valida-individuo
  ```

- Para verificar grupos:

  ```
  cd /var/nuestra-escuela/grupo
  make
  ```

- Para verificar planes de estudio:

  ```
  cd /var/nuestra-escuela/planest
  ```

6. Los `Makefile` y el programa **make** sólo verificarán archivos que hayan sido modificado tras la última verificación. Para lograrlo emplean la fecha de modificación de los archivos por verificar y de archivos vacíos usados con este propósito (por ejemplo después de verificar `planest/2007/col.planest` se creará o se actualizará la fecha del archivo `planest/2007/col.planestok`).

```
make
```

- Para verificar progreso de los estudiantes:

```
cd /var/nuestra-escuela/estudiante
make valida-prc
```

3.5. Generación de reportes

Comúnmente en un colegio se generan reportes para padres al finalizar cada periodo y al finalizar el año. Esporádicamente podría requerirse un certificado de calificaciones de un año anterior (con información de cierre de año). Labores que se realizan con el programa **reporte** y scripts que asisten su utilización.

En el directorio **reporte/2007** de la plantilla generada quedará un script y archivos para generar reportes periódicos en LaTeX. En el resto de esta sección se presenta como se generarían reportes empleado estas herramientas. Puede modificarlas de acuerdo a los requerimientos de su institución o si lo requiere puede crear un formato completamente diferente (por ejemplo en DocBook o HTML).

3.5.1. Modificaciones a las clases para LaTeX

Los formatos que sugerimos para registros valorativos de periodos (ver el presentado al comienzo del capítulo Sección 3.2) y finales se basan en clases para LaTeX:

`periodo.cls`
: que es empleada para registros valorativos de periodos,

`final.cls`
: que es empleada para registros valorativos de final de año y

`regval.cls`
: que es una clase común empleada por los dos primeros.

Entre las características de estas clases están:

- Generan PostScript para imprimir en papel tamaño oficio.
- Incluye datos de la institución (nombre, slogan, rector(a)).
- Incluye grado, periodo, moderador del grado, nombre de estudiante y observaciones.
- Incluye valoraciones por área y por indicador (en el caso del registro final suponemos que sólo se incluyen indicadores pendientes).
- Cuentan con espacio para la firma del moderador del grupo y el/la rector(a).

Modifíquelas de acuerdo a sus requerimientos, por ejemplo puede comenzar modificando el nombre de la institución, el eslogan y el nombre del rector en la clase `regval.cls`. Con algo de experiencia con LaTeX, con la herramienta **reporte**, y eventualmente modificando el script **imprep.sh** podrá adaptarla mucho más para su institución.

3.5.2. Reporte al final de un periodo

Una vez se han valorado todos los indicadores de un periodo pueden generarse reportes de un estudiante o de un grado.

Puede generar el reporte de un estudiante (digamos `robcar` de primero) con la siguiente secuencia de instrucciones (desde el directorio base):

```
cd reporte/2007
../imprep.sh bprimaria/g1 p2007-1 ../../estudiante/1980/robcar/ > robcar-2007-1.tex
```

con lo cual creará el archivo TeX `robcar-2007-1.tex` que podrá procesar para obtener un PostScript (`robcar-2007-1.ps`) con:

```
latex robcar-2007-1.tex
dvips -t legal -o robcar-2007-1.ps robcar-2007-1.dvi
```

e imprimir el PostScript con:

```
lpr robcar-2007-1.ps
```

Si desea especificar una fecha para el reporte indíquela como cuarto parámetro por ejemplo:

```
cd reporte/2007
../imprep.sh bprimaria/g1 p2007-1 ../../estudiante/1980/robcar/ 10/3/2007 > robcar-2007-1.te
```

Si en lugar de generar un archivo LaTeX desea generar XML (para ver la información disponible) puede emplear:

```
cd reporte/2007
XML=1 ../imprep.sh bprimaria/g1 p2007-1 ../../estudiante/1980/robcar/
```

Para generar los reportes de un grado (digamos `bprimaria/g1`) desde el directorio base ejecute:

```
cd reporte/2007
../imprep.sh bprimaria/g1 p2007-1
```

con lo cual generará un reporte en TeX de cada estudiante del grado `bprimaria/g1` en el directorio `/tmp/impresion` con nombres como `/tmp/impresion/repg1-1.tex`, `/tmp/impresion/repg1-2.tex` y así sucesivamente (uno por cada estudiante del grado). En caso de generar un grado, si lo requiere, puede especificar una fecha como tercer parámetro. Para imprimirlos

todos puede emplear un script como **genps.sh** (disponible en el directorio `reporte`) que genera PostScript de cada archivo con extensión `.tex` del directorio `/tmp/impresion`.

3.5.2.1. Computo de promedios

Los reportes de la plantilla creada por **sigue-colegio.sh**, emplean un programa escrito en lenguaje `prom` (ver intprom(1)) para calcular promedios por área. Tal programa incluye unas reglas genéricas que pueden modificarse fácilmente. Las reglas para calcular promedios por área periódicamente está en el archivo `periodo.prom` del directorio de reportes, el contenido de este archivo es:

```
/* Calcula 'promedio' con base en valoraciones de indicadores (E,B o I).
   Si más del 30% de los indicadores son I el promedio es deficiente.
   Si más del 10% de los indicadores son I el promedio es insuficiente.
   Si no hay indicadores I y el 90% o más son E el promedio es excelente.
   Si no hay indicadores I y entre el 70% (inclusive) y el 90% son E
   el promedio es sobresaliente.
   En el resto de casos el promedio es aceptable.
   Comportamiento es caso especial.

   Dominio público. 2003.
*/

aserción(ejecutor=="reporte");

numi=0;
numb=0;
nume=0;
tot=0.0;


si (id=="Comportamiento") {
        si (val_id[0]=="Comportamiento.Disciplina") {
                vdis=val[0];
                vcond=val[1];
        }
        sino {
                vcond=val[0];
                vdis=val[1];
        }
        si (vdis=="E" && vcond=="E") {
                retorna "Excelente";
        }
        si (vdis=="B" && vcond=="E") {
                retorna "Sobresaliente";
        }
        si (vcond=="B" || vcond=="E") {
                retorna "Aceptable";
        }
        retorna "Insuficiente";
}
```

```
max=tamaño(val);
i=0;
mientras (i<max) {
si (val[i]=="E") {
nume=nume+1;
}
si (val[i]=="B") {
numb=numb+1;
}
si (val[i]=="I") {
numi=numi+1;
}
i=i+1;
}
tot=numi+numb+nume;

si (tot==0) {
mensaje("Error calculando promedio: Faltan calificaciones\n");
retorna " ";
}
pi=numi/tot;
pb=numb/tot;
pe=nume/tot;

si (pi>0.3) {
retorna "Deficiente";
}
si (pi>0.1) {
retorna "Insuficiente";
}
si (numi==0.0 && pe>=0.9) {
retorna "Excelente";
}
si (numi==0.0 && pe>=0.7) {
retorna "Sobresaliente";
}
retorna "Aceptable";
```

Este programa es ejecutado por **reporte**, el cual entre otras (ver reporte(1)) pasa el vector `val` con las valoraciones de los indicadores de todas las asignaturas del área. Note que este programa trata el área Comportamiento de manera especial, para el resto de áreas cuenta la frecuencia de valoraciones excelente, las de bien y las de insuficiente y con base en los porcentajes de cada una devuelve una valoración para el área.

Cuando modifique o haga programas para calcular promedios puede resultarle de utilidad mostrar información de depuración por error estándar, por ejemplo:

```
  mensaje("El valor de a es" a "y el de b es " b);
```

También podrá verificar los tipos de su programa prom (suponiendo que retornan un dato de tipo `cadena`) con:

```
  intprom -C cadena -i periodo.prom
```

O antes de usar sus reglas para calcular promedios con **reporte** puede probar con valores inicializados por usted en un prologo (lo cual típicamente sería función de **reporte**) por ejemplo:

```
intprom -prologo "ejecutor=\"reporte\"; val[0]=\"E\"; val[1]=\"B\";" periodo.prom
```

3.5.3. Reporte al final de un año

Al concluir un año de acuerdo al decreto 230 de 2002 deben computarse promedios por área. Para esto se sugiere usar un nuevo periodo (por ejemplo `p2007-5`) en el cual se agregan valoraciones finales a los archivos de progreso de cada asignatura de cada estudiante, así como unas reglas para computar los promedios por asignatura y unas reglas para computar promedios de área a partir de promedios en asignaturas (ver Sección 3.5.2.1) .

> **Aviso**
>
> En los archivos del plan de estudio, no deben agregarse indicadores de logro nuevos en el periodo añadido para reportes finales.

Para insertar los promedios en los archivos de progreso de cada asignatura puede emplearse la herramienta **iniperiodo**, por ejemplo así:

```
cd planest/2007
iniperiodo -anota "Fallas:0;Observaciones:" -tipoval Indicador \
  -prom ../../reporte/2007/asig-final.prom -prom_rango "p2007-1:p2007-4" \
  -norec "Recuperable;no" -p p2007-5 -pendiente "I" col.planest
```

o más breve empleando el script **planest/inifinal.sh**:

```
cd planest/2007
../inifinal.sh 2007
```

Con esto se calculan promedios de las valoraciones más recientes de todos los indicadores valorados entre los periodos p2007-1 y p2007-4, las reglas para promediar están en `../../reporte/2007/asig-final.prom` y los promedios se agregan en cada asignatura en el periodo p2007-5.

Antes de modificar los archivos de progreso, puede experimentar el computo sin agregar valoraciones a los archivos empleando la opción `-noescribir` de **iniperiodo** (eventualmente junto con las ayudas sugeridas en la sección anterior para depurar programas `prom`). En caso de requerir sobreescribir promedios con reglas nuevas puede emplear la opción `-remplazar`.

Para generar reportes finales también puede usar el script **imprep.sh** antes presentado, por ejemplo:

```
cd reporte/2007
../imprep.sh bprimaria/g1 p2007-5
```

al generar para el periodo p2007-5, este script usará `area-final.prom` para computar promedios con base en las valoraciones finales por asignatura[7]. Este script también empleará para el último periodo la clase de LaTeX `final.cls`[8]

3.6. Interfaz PHP para profesores

De seguir las sugerencias presentadas en este capítulo (en cuanto a información mínima por almacenar, estructura de directorios, identificadores y permisos) y de contar con servidor Apache y PHP en el mismo computador donde está la información, será posible que los profesores ingresen valoraciones durante cada periodo con un navegador y una interfaz sencilla.

En el momento de este escrito esta interfaz cuenta con dos mecanismos posibles de autenticación: (1) empleando un archivo propio de usuarios y claves o (2) empleando mecanismos de autenticación del sistema operativo. De resultarle posible sugerimos el que emplee la autenticación del sistema porque le permite conservar el esquema de permisos que sugerimos (ver Sección 3.3.2).

Además de los mecanismos mencionados (que asignan una clave a cada profesor) sugerimos emplear autenticación básica de Apache para ingresar al directorio de la interfaz con un usuario y una clave conocida sólo por todos los profesores.

3.6.1. Preparación

3.6.1.1. Preparación del sistema, de Apache y de PHP

Puede mejorar seguridad en el directorio donde instalará la interfaz usando archivos `.htaccess` (hay uno en el directorio donde instale la interfaz y otro en el subdirectorio `auth`). Para usarlos debe permitir a Apache sobrecargar algunos parámetros para el directorio donde instale la interfaz y para el subdirectorio `auth`. Además sugerimos emplear autenticación básica (con un usuario y una clave conocida por todos los profesores) y eventualmente restringir uso por IP. Para lograr estos propósitos verifique que en el archivo de configuración de Apache en la sección que corresponde al directorio en donde estará la interfaz PHP (digamos `/var/www/ssl/sigue`) diga:

```
<Directory /var/www/ssl/sigue/>
   AllowOverride Options AuthConfig Limit
</Directory>
```

y de requerirlo que en la sección donde se carguen módulos se cargue el módulo de autenticación, por ejemplo:

```
LoadModule auth_module /usr/lib/apache/1.3/mod_auth.so
```

7. A diferencia de `asig-final.prom` que realiza promedios sobre valoraciones de indicadores del periodo.
8. En el caso de periodos, `imprep.sh` emplea la clase `periodo.cls`.

El archivo `.htaccess` que se distribuye implementa autenticación básica, para esto emplea el archivo de claves `auth/sigue` que se distribuye sólo con el usuario `sigue` con clave `sigue`

> **Aviso**
>
> Si va a emplear autenticación básica para el directorio donde estará la interfaz PHP, recuerde cambiar el usuario y la clave periódicamente (con `htpasswd`) e informar a todos los profesores.

En el archivo de configuración de Apache verifique que también se emplee `index.php` en la directiva `DirectoryIndex`.

Con respecto a PHP se requiere haber instalado soporte para XML. En `php.ini`se sugiere en general desactivar `register_globals` —aunque esto también lo hará el archivo `.htaccess` que se instala junto con la interfaz— y aumentar tiempos máximos de ejecución y para ingreso de datos por ejemplo:

```
max_execution_time = 900
max_input_time = 900
```

Recuerde reiniciar Apache después de modificar el archivo de configuración de este servidor o el de PHP.

3.6.1.1.1. Preparación de la interfaz PHP

Con el script **sigue-intcol.sh** instale los archivos de la interfaz PHP en un directorio que pueda accederse con un navegador.

> **Aviso**
>
> Se recomienda que el directorio en el que se instale la interfaz PHP sea servido con SSL y además cuente con autenticación básica.

Por ejemplo si los datos del colegio están en `/var/nuestra-escuela`, el propietario y grupo del proceso con el que corre Apache es `www:www` y desea instalar la interfaz en el directorio `/var/www/ssl/sigue` ejecute:

```
cd /var/www/ssl
sigue-intcol.sh sigue "www:www" /var/nuestra-escuela
```

El script `sigue-intcol.sh` creará el directorio, compilará un programa de autenticación y de ejecutarse desde la cuenta root, establecerá permisos seguros[9]. Si su sistema no es soportado se empleará un archivo de claves propios como método de autenticación, aunque eventualmente usted mismo podrá ejecutar otra regla del Makefile para emplear otro método de autenticación como se explica en las siguientes secciones (o tal vez un método nuevo, escrito para su caso particular)[10].

9. Note que si usa autenticación del sistema, el programa de autenticación debe correr con suid de root para poder escribir en los archivos de los profesores en el directorio de datos. Hemos procurado hacerlo seguro, por favor reporte eventuales fallas en nuestro seguidor público de fallas de seguridad.
10. Por ejemplo para compilar el programa que autentica con base en el archivo de claves propias: `cd /var/www/ssl/sigue/auth make claves-propias`

Finalmente edite el archivo `auth/confcol.php` y además de verificar la información existente cambie el valor de la variable `PALABRA_SITIO` con una palabra secreta[11].

Si al emplear uno de los métodos de autenticación que se describen a continuación tiene problemas, revise la bitacora de errores de Apache, pues allí quedan los mensajes de error que producen los programas de autenticación.

3.6.1.1.2. Autenticación con archivo de claves propio

Cada profesor debe tener un login (referenciado en el archivo de datos del profesor) y una clave. Esta información debe registrarse en un archivo, el cual puede crearse después de instalar la interfaz PHP (como se explica en la siguiente sección) con el nombre `auth/usuarios`. Para crearlo y manejarlo emplee la herramienta **htpasswd** distribuida con Apache.

Para que todos los profesores puedan escribir en los archivos estos deben tener permisos que dejen escribir al usuario bajo el cual corra Apache.

> **Aviso**
>
> Este método de autenticación no es tan seguro, porque requiere debilitar el esquema de permisos de los datos del colegio. De ser posible emplee autenticación del sistema operativo junto con el esquema de permisos que hemos sugerido.

Puede compilar desde el directorio `auth` con

```
make METODO=claves-propias
```

recuerde modificar la variable `AUTHPGM` del archivo `auth/confcol.php` para que quede:

```
$GLOBALS['AUTHPGM']="auth/claves-propias";
```

3.6.1.1.3. Autenticación del sistema operativo

En el momento de este escrito sólo se soporta autenticación local con OpenBSD y con archivo de claves Shadow (como en Linux Debian).

El servidor web no debe correr con **chroot** para tener acceso a los archivos de claves del sistema (en el caso de OpenBSD esto se desactiva con `httpd_flags="-u"` en `/etc/rc.conf.local`). Para que el servidor web pueda leer los datos del colegio debe agregar el usuario bajo el cual corre el servidor al grupo `profesor`. Por ejemplo en OpenBSD el archivo `/var/www/conf/httpd.conf` debe modificarse para que incluya[12]:

```
User www
Group www
```

11. La palabra secreta que configure será empleada para encriptar algo de la información de la sesión PHP.
12. En el caso de Linux Debian el usuario es `www-data` y el grupo es `www-data`.

Después el usuario bajo el cual corre el servidor se agrega al grupo `profesor` (por ejemplo editando `/etc/group`).

> **Aviso**
>
> Verifique en los archivos de configuración de Apache, que este no permitirá ver directorios de lectura exclusiva para el grupo `profesor`.

Dado que la autenticación debe hacerse con un programa con suid root (por ejemplo en el caso de OpenBSD `auth/openbsd-local`), verifique que el directorio donde quedará la interfaz pueda ejecutar programas suid root (por ejemplo que entre las opciones para montarlo en `/etc/fstab` no esté `nosuid`).

Puede compilar desde el directorio `auth` bien con

```
make METODO=openbsd-local
```

o bien con

```
make METODO=shadow-local
```

recuerde establecer el que eliga en la variable `AUTHPGM` del archivo `auth/confcol.php`

3.6.1.1.4. Autenticación con directorio LDAP

Si el colegio cuenta con un servidor LDAP que incluya información para autenticar puede emplear este mecanismo de autenticación. Puede consultar como configurar uno en http://dhobsd.pasosdejesus.org/?id=Autenticaci%F3n+con+OpenLDAP

Con respecto al servidor web, tenga en cuenta las mismas recomendaciones del caso anterior Sección 3.6.1.1.3

Copie la plantilla `auth/cmdldap.h.plantilla` en `auth/cmdldap.h` y modifiquela para que tenga:

- El URI del servidor LDAP.
- El DN de la organización.
- El DN de las personas de la organización.

Por defecto la autenticación LDAP empleará como filtro `(&(objectclass=posixAccount)(uid=`*`login`*`))`, pero si este no es un filtro apropiado para su caso modifiquelo en las fuentes de `cmdldap.c`.

Finalmente compile desde el directorio `auth` con:

```
make METODO=cmdldap
```

y establezca la variable `AUTHPGM` del archivo `auth/confcol.php` para que quede:

```
$GLOBALS['AUTHPGM']="auth/cmdldap";
```

3.6.1.1.5. Deshabilitar autenticación para efectuar pruebas

Posiblemente el/la administrador(a) requiere ingresar como un profesor para verificar funcionamiento de la interfaz PHP. En tal caso puede compilar el método `sinclaves`:

```
make sinclave
```

Y podrá configurar este método de autenticación (que acepta cualquier clave) cambiando `auth/confcol.php` para que la variable `AUTHPGM` sea:

```
$GLOBALS['AUTHPGM']="auth/sinclaves";
```

> **Aviso**
>
> Este método permite acceso a cualquier profesor, con cualquier clave. Es sólo para probar la interfaz, habilite un método de autenticación real (como los descritos en las secciones anteriores) tan pronto complete sus pruebas. Asegurese también de no permitir cambios no autorizados a `auth/confcol.php` ni en general a las fuentes de `sigue` quitando permiso de escritura a todos los usuarios excepto el administrador.

3.6.2. Uso de la interfaz

3.6.2.1. Uso por parte del administrador

Los estados de la interfaz dependen del cronograma, de la fecha y de las variables `VALORAR` y `ACTIVA` del archivo `auth/confcol.php`. El cronograma y la fecha del computador determinarán el periodo actual. `VALORAR` indica si pueden o no hacerse valoraciones del periodo actual. `ACTIVA` indica si la interfaz es utilizable o no por profesores.

Al principio de un periodo la variable `VALORAR` debe ser `false`, mientras que `ACTIVA` debe ser `true`, para que los profesores puedan editar logros e indicadores del periodo y futuros pero que NO puedan modificar valoraciones, al llegar a cierta fecha del periodo conocida por profesores y administrador, no se aceptarán más ediciones a logros e indicadores. El administrador desactivará temporal y rápidamente la interfaz (asignando `false` a `ACTIVA`) para ejecutar `iniperiodo` (ver Sección 3.4.3).

Una vez se completé la ejecución de `iniperiodo` (la cual prepara todos los archivos de progreso de estudiantes), el administrador pondrá la variable `VALORAR` en `true` y activará la interfaz PHP para que los profesores puedan ingresar las valoraciones del periodo.

Una vez se cumpla el tiempo de ingresar valoraciones y de efectuar recuperaciones, el administrador pondrá la variable `VALORAR` en `false`, dejando así todo listo para comenzar el siguiente periodo y para poder imprimir reportes del periodo concluido.

3.6.2.2. Uso por parte de profesores

Cuando la interfaz esté habilitada por el administrador, los profesores podrán ingresar con su clave para: (1) Editar logros e indicadores futuros o del periodo actual al comienzo del periodo, (2) cuando sea el momento acordado podrán tanto valorar indicadores y logros del periodo como recuperar indicadores pendientes (i.e indicadores con valoración `I`).

- Cada profesor podrá entrar a la interfaz empleando su clave (de acuerdo al método de autenticación implementado por el administrador, bien la clave en el sistema o bien la clave particular para esta interfaz).

- Podrá elegir la materia y el curso que desea editar (sólo entre las que dicta).

- Podrá añadir o eliminar indicadores y logros de uno o más periodos, así como referencias a indicadores en algún periodo (si un indicador es no recuperable pueden agregarse referencias a este en más de un periodo).

- En el caso de valoraciones del periodo actual, podrá ingresar valoraciones de cada indicador y número de fallas de cada estudiante y eventuales observaciones (que se acumularan con observaciones que otros profesores hagan —por consistencia de la información final, puede ser buena política que sólo en una asignatura como disciplina o convivencia se ingresen las observaciones). En el caso de recuperaciones podrá también elegir el periodo que desea editar y modificar sólo las valoraciones pendientes.

Apéndice A. Créditos

Ayudaron con sus sugerencias y correcciones al menos (por favor recuerdenos si olvidamos incluir su nombre):

- Ruben Amortegui (browser80@users.sourceforge.net)
- Isamary García (is-garci@users.sourceforge.net)

Los autores son:

- Igor Támara (ikks@users.sourceforge.net)
- Jaime Irving Dávila (jaimeirvingd@users.sourceforge.net)
- Vladimir Támara (vtamara@users.sourceforge.net)

Además de nuestro código original, `sigue` incluye porciones de las siguientes fuentes de dominio público:

- Manejo de línea de comandos en script de configuración basado en código de WWWeb Tide Team: http://www.ebbtide.com/Util/ksh_parse.html
- Esquema de configuración de `repasa` y ayudas para documentación en DocBook de `repasa` (http://structio.sourceforge.net/repasa)

Cómo fuente de inspiración o de ideas se emplearon (lista con seguridad incompleta):

- Experiencia ganada empleando versiones de desarrollo de `sigue` desde 2002 en el colegio colombiano Gimnasio Fidel Cano (http://www.gfc.edu.co).
- Legislación educativa colombiana vigente, e.g Plan decenal de educación 1994-2004. Ley 115/1994. Structio mantiene un archivo (http://structio.sourceforge.net/leg) de este tipo de legislación.
- Para la impresión en LaTeX, ideas del macro `\edefappned` de eplain http://www.ccs.neu.edu/home/dorai/eplain/
- La idea de emplear XML y buscar generalidad ha sido tomada de EduML.
- Para verificar configuración segura de PHP se tomó código de erfurtwiki (http://erfurtwiki.sourceforge.net)

Apéndice B. Derechos

Nosotros Igor Támara, Jaime Irving Dávila y Vladimir Támara autores del programa `sigue` y su documentación. Por este medio cedemos las fuentes del programa y de la documentación al dominio público, renunciando a todos los derechos patrimoniales con lo que esperamos facilitar su adaptación y uso especialmente en colegios. En particular quedan permitidos para siempre: el uso, la copia, la redistribución y la modificación de este escrito y sus fuentes.

Confirmamos que el trabajo es de nuestra autoría y que hemos dado crédito por las partes que hemos copiado o adaptado de otras obras (ver Apéndice A).

Cedemos este trabajo al dominio público, en el marco de la legislación colombiana, de acuerdo al artículo 188 de la ley 23 de 1982.

Dado que `sigue` ha sido desarrollado por voluntarios, y que no hemos cobrado a quienes lo han obtenido, no ofrecemos garantía ni reembolso de tipo alguno.

Apreciamos que para dar crédito se cite: http://structio.sourceforge.net/sigue Queremos recordar que de basarse en las fuentes de `sigue` debe respetar los créditos para no incurrir en plagio (ver http://structio.sourceforge.net/vladimir/dominio_publico.html).

Si desea ayudar a mejorar `sigue` o desea obtener las fuentes puede visitar en Internet: http://structio.sourceforge.net/sigue.

Apéndice C. Instalación

En el sitio de desarrollo de `sigue` en Internet: http://structio.sourceforge.net/sigue es posible obtener las fuentes, así como un paquete precompilado para sistemas OpenBSD en plataforma x86. Este paquete puede instalarse de la manera estándar en sistemas OpenBSD (con **pkg_add**).

A continuación se presentan instrucciones para compilar las fuentes en sistemas tipo Unix.

Requerimientos

- Un sistema tipo Unix (Se ha probado especialmente con OpenBSD http://www.openbsd.org).
- Ocaml 3.x (se ha probado especialmente con 3.07). Puede obtener las fuentes en: http://caml.inria.fr/ocaml/. Hay paquetes precompilados de este lenguaje para diversas plataformas (e.g OpenBSD, Debian, Windows).
- **make**, **sed**, **grep** y un interprete de comandos (e.g **pdksh**, **bash**).
- La librería `Markup` instalada. Está disponible en: http://www.ocaml-programming.de/markup/
- El programa `repasa` instalado. Está disponible en: http://structio.sourceforge.net/repasa
- Opcional: Corrector de ortografía **ispell** y diccionario en español.
- Opcional: Si desea emplear la interfaz PHP, requerirá el servidor web Apache (>=1.3) preferiblemente con SSL y PHP (>= 4.1) en el computador donde está la información. Se ha probado especialmente con Apache 1.3.27 y PHP 5
- Opcional: Para respaldar y tener la posibilidad de rastrear modificaciones se recomienda emplear CVS.

Compilación

Primero desempaquete y configure con:

```
tar xvfz sigue-1.2.tar.gz
cd sigue-1.2
./conf.sh
```

si desea especificar una ruta de instalación (por defecto es `/usr/local`), emplee el parámetro `-p` del script **conf.sh**, por ejemplo:

```
./conf.sh -p /home/juan/
```

Este proceso buscará los programas requeridos en algunos sitios y hará algunas pruebas. Si algún componente requerido no es encontrado usted podrá indicar la ubicación cuando **conf.sh** lo solicite o puede cancelar la configuración y editar directamente **confv.sh** para después ejecutar nuevamente **conf.sh**.

Cuando complete la configuración compile con:

```
make
```

o si Ocaml en su plataforma soporta compilación nativa:

```
make opt
```

Pruebas de regresión

Permiten verificar que los programas que compiló pasan diversas pruebas en su plataforma. Aunque no es indispensable ejecutarlas antes de instalar, es sugerido. Después de compilar puede ejecutarlas desde el directorio de fuentes con:

```
cd regr
./test.sh
```

o si compiló nativamente:

```
cd regr
./test.sh -opt
```

Si alguna prueba falla la ejecución se detendrá para indicarlo, y deberá presionar **ENTER** para continuar. Después de completar estas pruebas, tanto en caso de que fallen como en caso de que no fallen solicitamos enviarnos el archivo `test.log` en un correo dirigido a <`structio-info@lists.sourceforge.net`>. Así podremos determinar en que plataformas se ha podio compilar `sigue` (para soportarla mejor) y si hubo fallas podremos corregirlas e informarle.

Instalación

Una vez compile puede instalar en algunos directorios dentro del directorio prefijo (especificado al configurar) con:

```
make instala
```

o

```
make instala.opt
```

Si conserva las fuentes o las configura de la misma forma que al instalar, podrá desinstalar posteriormente con:

```
make desinstala
```

Para conservar las fuentes ahorrando espacio (eliminando código objeto producido durante la compilación y binarios no instalados):

```
make limpia
```

o para eliminar también la configuración que haya hecho:

```
make limpiadist
```

Documentación

Puede consultarla o descargarla de Internet en diversos formatos desde:
http://structio.sourceforge.net/sigue

Junto con las fuentes de `sigue` se incluyen:

- Documentación para usuario en formato HTML en el directorio `doc/usuario/html`
- Fuentes en DocBook de la documentación para usuario en el directorio `doc/usuario`. Para generarla consulte `doc/usuario/Leame.txt`. Si colabora en el desarrollo puede generar diversos archivos de documentación para usuarios (e.g archivo Instala.txt, Derechos.txt, documentación de `doc/usuario`) con **make doc**.
- Páginas `man` de los formatos y programas, que por defecto son instaladas al instalar `sigue`.
- Si emplea Ocaml 3.06 o posterior y desea generar la documentación técnica de las fuentes en HTML, después de compilar ejecute:

```
mkdir doc/technical
mkdir doc/technical/srcdoc
make srcdoc
```

I. Documentación de referencia de formatos

individuo

Nombre

individuo — Formato de archivo: datos de un individuo

Descripción y ejemplos

Pueden mantenerse diversos datos de un individuo como se ejemplifica a continuación:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE individuo PUBLIC "-//Structio//DTD individuo 1.0" "individuo.dtd">
<individuo mindatos="Apellidos;Nombres;TipoDocumento:d='TI'||d='CC';Documento:d>0;Foto">

   <dato tipo="Apellidos">Perez</dato>
   <dato tipo="Nombres">Juan</dato>
   <dato tipo="TipoDocumento">TI</dato>
   <dato tipo="Documento">1233</dato>
   <dato tipo="Login">juaper</dato>
   <dato tipo="Foto">fotos/juaper.jpg</dato>

</individuo>
```

Note que

- Cada dato se especifica en un elemento `dato`
- El tipo de cada dato se especifica en el atributo `tipo` del elemento dato.
- Los tipos de datos que deben aparecer en el archivo y restricciones sobre estos se especifican con el atributo `mindatos` del elemento raíz.

Un archivo con datos de un individuo puede chequearse con el programa **sigchq**.

Descripción detallada de cada elemento

`individuo` - Raíz de progreso en datos de individuo

El elemento raíz `individuo` puede emplear los siguientes atributos

mindatos

que especifica los datos mínimos que deben aparecer en el archivo (separando uno de otro con ';'). Pueden especificarse restricciones al valor de cada dato usando predicados sobre la variable `d` (ver sección sobre predicados en prcla(5)). En el ejemplo presentado el tipo `Documento` tiene asociado el predicado `d>0`. **sigchq** revisará que el dato de tipo `Documento` cumpla este predicado.

formato

que especifica el formato en el que se escriben anotaciones (ver ,clasif(5)).

Los subelementos deben ser `dato`, y debe haber al menos datos con los tipos especificados en `mindatos`, aunque puede haber otros.

`dato` - Un dato del individuo

Con este elemento se especifica un dato del individuo, el tipo de dato se especifica en el atributo `tipo`.

Ver también

sigchq(1),

`grupo`

Nombre

`grupo` — Formato de archivo: grupo de individuos

Descripción y ejemplos

Puede mantenerse un grupo de individuo como se presenta en el siguiente ejmplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE grupo PUBLIC "-//Structio//DTD grupo 1.0" "grupo.dtd">
<grupo>
```

```
  <desc>Primero</desc>

  <refind arch="juaper.ind"/>
  <refind arch="cammen.ind"/>

</grupo>
```

Un archivo con un grupo puede chequearse con el programa **sigchq**.

Descripción detallada de cada elemento

`grupo` - Raíz de archivo para grupos

A continuación del elemento raíz `grupo` debe especificarse una descripción con el elemento `desc`. Después pueden emplearse los subelementos `refind` para referenciar un individuo, o `anota` para agregar anotaciones (ver página man de elementos-comunes(5)).

`refind` - Referencia a un individuo.

Con este elemento se referencia el archivo con datos de un individuo. El archivo se especifica en el atributo `arch`, debe ser un archivo que siga el DTD para individuos (ver individuo(5)). Los archivos de individuos referenciados no deben estar repetidos.

Ver también

sigchq(1), individuo(5),

`prcla`

Nombre

`prcla` — Formato de archivo: progreso en estudio de clasificaciones

Descripción y ejemplos

El progreso que un estudiante hace al estudiar clasificaciones, puede registrarse con este formato. De cada clasificación puede registrarse progreso en ejercicios y una valoración en algún periodo. Por ejemplo suponiendo que el archivo de clasificaciones (ver clasif(5)) defina los indicadores `contexto.geografía`, `contexto.historia` y `establece.reflexiona` el archivo de progreso podría ser:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE prcla PUBLIC "-//Structio//DTD progreso en clasificaciones 1.0" "prcla.dtd">

<prcla conceptos=":(g='S' || g='B' || g='A' || g='D' || g='I');Indicador:(g='E' || g='B' ||
  <progcla idcla="ejsimple.cla:contexto.historia" tipo="Indicador">
    <val concepto="I" periodo="p2004-1"/>
    <val concepto="B" periodo="p2004-2"/>
  </progcla>
  <progcla idcla="ejsimple.cla:contexto.geografía" tipo="Indicador">
    <val concepto="I" periodo="p2004-1"/>
    <val concepto="I" periodo="p2004-2"/>
  </progcla>
  <progcla idcla="ejsimple.cla:establece.reflexiona" tipo="Indicador">
    <val concepto="I" periodo="p2004-2"></val>
    <progejer idejer="establece.reflexiona.e1" fecha="1/1/2004" concepto="I"/>
    <progejer idejer="establece.reflexiona.e1" fecha="10/3/2004" concepto="I"/>
  </progcla>
  <anota valor="p2004-1" tipo="Fallas">3</anota>
  <anota valor="p2004-2" tipo="Fallas">10</anota>
  <val concepto="D" periodo="p2004-3"></val>
</prcla>
```

Note que:

- Los conceptos posibles se indican en el atributo `conceptos` del elemento `prcla`, para indicadores pueden ser `E`, `B` e `I`, mientras que los generales pueden ser `S`, `B`, `A`, `D`, `I`
- Los indicadores `contexto.historia` y `contexto.geografía` se valoran en los dos periodos (con el elemento `val`), mientras que el indicador `establece.reflexiona` se valora en un periodo (`p2004-2`).
- Hay una valoración general no asociado a Indicador alguno, para el periodo `p2004-3`

Un archivo de progreso en clasificaciones puede chequearse con el programa **sigchq**.

Descripción detallada de cada elemento

`prcla` - Raíz de progreso en clasificaciones

El elemento raíz `prcla` puede emplear los siguientes atributos

`periodos`

que define los periodos posibles, se especifican como un predicado (ver sobre predicados a continuación) en la variable `p`.

`conceptos`

que define los conceptos posibles, los cuales se especifican con un predicado en la variable `c`

`formato` y `longitud`

que especifica el formato en el que se escriben anotaciones (ver ,clasif(5)).

Como subelementos puede tener anotaciones (`anota`, ver página man elementos-comunes(5)), valoraciones generales (`val`) y progreso en clasificaciones (`progcla`). De haber anotaciones deben emplear el atributo valor con el periodo al cual aplican.

Predicados para especificar conceptos y periodos

Un predicado es una expresión que debe evaluar a verdadero o falso. La expresión puede incluir constantes (números o cadenas), una variable (por ejemplo `p` en el caso de periodos o `c` en el caso de conceptos), operadores de relación o operadores booleanos. A continuación se ejemplifican las constantes y los operadores de relación operando con la variable `c`:

Tabla 1. Operadores de relación

Operador	Descripción
`c='I'`	Es verdadero si el concepto (i.e la variable `c`) es `I`
`c<5`	Verdadero si `c` es menor al entero 5.
c>2.3	Verdadero si `c` es mayor a 2.3
`c>='Agua'`	Verdadero si `c` es lexicográficamente mayor o igual a la cadena `Agua`2.3
`c<=5`	Verdadero si `c` es menor o igual al entero 5.
`c!=0`	Verdadero si `c` es diferente a 0.

Operador	Descripción
`c~='p[0-9]+-[0-9]'`	Verdadero si `c` concuerda con la expresión regular dada. Por ejemplo la cadena 'p2001-2' concuerda, pero 'p-1' no. La expresión regular tiene la sintaxis de las expresiones regulares de Ocaml. De acuerdo a la documentación del módulo Str, puede emplear los caracteres especiales `$^.*+?[]` así: `.` concuerda con todo caracter, `*` posfijo concuerda con la expresión que lo precede cero o más veces, `+` posfijo concuerda con la expresión que lo precede más de una vez, `?` posfijo concuerda con la expresión que lo precede cero o una vez, `[...]` concuerda con un conjunto de caracteres pueden especificarse rangos con `-` o complementos comenzando con `^`, `^` concuerda con el comienzo, `%` concuerda con el final, `|` infijo representa alternativa entre dos expresiones.

Los operadores booleanos se ejemplifican a continuación:

Tabla 2. Operadores booleanos

Operador	Descripción
!(c=='x')	Negación.
c<2 || c>4	Disyunción (operador 'o').
c!=2 && c!=3	Conjunción (operador 'y').

Además de una sintaxis correcta, un predicado debe ser bien tipado, es decir el uso de la variable siempre se da con un sólo tipo de datos (bien enteros o bien sólo flotantes o bien sólo cadenas).

`progcla` - Progreso en una clasificación

La clasificación debe especificarse en los atributos `idcla` y `tipo` con la respectiva identificación y tipo. La identificación debe comenzar con el nombre del archivo seguida de ':' y de la identificación de la clasificación (no hay archivo de clasificación por defecto en el caso de archivos de progreso en clasificaciones).

Sus subelementos pueden ser valoraciones (elemento `val`), ejercicios de la clasificación (elemento `progejer`), otras referencias a clasificaciones anidadas dentro de la clasificación que este clasifica o anotaciones.

`progejer` - Progreso en un ejercicio

Con este elemento se registra un concepto (atributo `concepto`) para un ejercicio desarrollado en una fecha (atributo `fecha`). El ejercicio referenciado debe ser un ejercicio de la clasificación en la que

aparece. La identificación del ejercicio debe estar en el atributo `idejer`, puede ser de la forma *`archivo`*:*`id`* y en tal caso el archivo debe corresponder al archivo de la clasificación en la que está, o de la forma *`id`* con lo que se supone que el archivo es el mismo de la clasificación en la que está.

`val` - Valoración

Este elemento registra una valoración en un periodo para la clasificación en la que está o si no está en clasificación alguna, se trata de una valoración global. Sus atributos son `periodo` y `concepto`. En caso de haber especificado predicados para periodos o conceptos (ver elemento `prcla`), el valor de estos atributos debe cumplir el predicado.

Ver también

sigchq(1),

Historia

Las valoraciones en referencias a clasificaciones se introdujeron para permitir valoración de indicadores, las referencias anidadas para permitir valoración tanto de logros como de indicadores, las valoraciones generales para permitir valoración general de una materia.

`planest`

Nombre

`planest` — Formato de archivo: plan de estudios de un colegio

Descripción y ejemplos

Puede mantenerse un plan de estudios, en particular uno que este de acuerdo a la legislación colombiana vigente (artículo 79 de la ley 115 de 1994 (http://structio.sourceforge.net/leg/1994ley115.html#A79)). A continuación se presenta un ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE planest PUBLIC "-//Structio//DTD plan de estudios 1.1" "planest.dtd">
<planest tipos_curso="Nivel;Grado" tipos_asignatura="Área;Asignatura">

  <desc>Ejemplo de plan de estudios</desc>
```

```
  <derechos tipo="Dominio público" tiempo="2003">Cedido al dominio público</derechos>
  <autor fecha="2002">structio-info@lists.sourceforge.net</autor>

  <curso tipo="Nivel" id="primaria"><desc>Primaria</desc>
    <curso tipo="Grado" id="g4" refgrupo="ejsimple.grp" refmoderador="lucpue.ind"><desc>4</d
      <asignatura tipo="Área" id="religión"><desc>Educación religiosa</desc>
        <asignatura tipo="Asignatura" id="Religión" refsectemas="ejsimple.sec"
              intensidad="2" prc="ejsimple.prc" refmoderador="casbel.ind">
              <desc>Religión</desc>
        </asignatura>
      </asignatura>
    </curso>
  </curso>
</planest>
```

Note que:

- Pueden especificarse jerarquías en los grupos de estudiantes con el elemento `curso` (en este ejmplo Nivel y Grado)
- Pueden especificarse jerarquías para las asignaturas con el elemento `asignatura` (en el ejemplo Área y Asignatura).
- Es posible mantener la carga académica de profesores en sus asignaturas (con el atributo intensidad). Y los elementos `restriccion` permiten especificar restricciones en el horario de algún profesor

Un archivo con un plan de estudios puede chequearse con el programa **sigchq**.

Descripción detallada de cada elemento

`planest` - Raíz del plan de estudios

El elemento raíz `planest` puede emplear los siguientes atributos

`tipos_curso`

Define los tipos posibles para cursos.

`tipos_asignatura`

Define los tipos posibles para asignaturas.

`formato` y `longitud`

que especifica el formato en el que se escriben anotaciones (ver clasif(5)).

`maneja`

que especifica el login en el sistema de quien maneja el plan de estudios, opcionalmente seguido de : y el grupo.

Debe tener una descripción especificada con el elemento `desc`, y puede tener a continuación los elementos comunes `anota`, `derechos`, `biblio` y `autor` (ver información sobre elementos comunes de `repasa`) o la especificación del plan de estudios para un curso iniciada con el elemento `curso` o información del horario con `restriccion`.

`curso` - Nivel o cursos

Permite especificar un grupo de estudiantes. Por ejemplo un nivel o un grado. El tipo de grupo se especifica en el atributo `tipo`. Si se requiere puede especificarse un archivo para el grupo con el atributo `refgrupo` (ver grupo(5)) y/o un moderador en el atributo `refmoderador` (ver individuo(5)).

`asignatura`

Permite especificar una asignatura. El tipo en el atributo `tipo`, la secuencia de temas en `refsectemas`, el moderador en `refmoderador`, la intensidad en `intensidad` y el nombre del archivo de progreso que se usará en `prc`.

Note que puede tener subelementos `restriccion` que permiten especificar restricciones para la asignatura en el horario.

`restriccion`

Permite especificar una restricción para el horario de una asignatura en un grado. El tipo en el atributo `tipo` (por ejemplo fija), la identificación de la hora a la que se aplica la restricción en el atributo `hora`. Puede incluirse una anotación en el atributo `anota`

Ver también

sigchq(1),

`rep`

Nombre

`rep` — Formato de archivo: reporte de progreso

Descripción y ejemplos

Un reporte de progreso se organiza como una lista de ítems con datos, como se presenta en el siguiente ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rep PUBLIC "-//Structio//DTD reporte 1.1" "rep.dtd">
<rep formato="texto">
  <dato tipo="Nombres">Juan</dato>
  <dato tipo="Apellidos">Perez</dato>
  <dato tipo="Documento">1233</dato>
  <dato tipo="Fecha">20/4/2002</dato>
  <dato tipo="login">juaper</dato>
  <dato tipo="Periodo">p2001-1</dato>
  <dato tipo="Grado">g1</dato>
  <dato tipo="computo">Deficiente</dato>
  <item tipo="Grado">
    <dato tipo="Moderador.nombre">Garcilazo</dato>
    <dato tipo="desc">Grado 1</dato>
    <dato tipo="Moderador.login">garpue</dato>
    <dato tipo="id">g1</dato>
    <dato tipo="Moderador.apellidos">Puentes</dato>
    <item tipo="Área">
      <dato tipo="desc">Educación ética y en valores humanos</dato>
      <dato tipo="id">Ética</dato>
      <item tipo="Asignatura">
        <dato tipo="Moderador.nombre">Justo</dato>
<dato tipo="Moderador.apellidos">Aguirre</dato>
        <dato tipo="Moderador.login">jusagu</dato>
        <dato tipo="intensidad">1</dato>
        <dato tipo="id">dh</dato>
        <dato tipo="desc">Dignidad humana</dato>
        <dato tipo="Fallas">3</dato>
<dato tipo="Observacion"></dato>
        <dato tipo="computo">Deficiente</dato>
        <item tipo="Logro">
          <dato tipo="id">dh.cla:vida</dato>
  <dato tipo="desc">Respeta la vida y pide respeto por la vida</dato>
          <item tipo="Indicador">
            <dato tipo="id">dh.cla.cla:vida.busca</dato>
    <dato tipo="desc">Busca posibles responsables de asesinato(s).</dato>
            <dato tipo="val">I</dato>
          </item>
          <item tipo="Indicador">
            <dato tipo="id">m1-2001.cla:vida.pide</dato>
    <dato tipo="desc">Confirma directamente con el presunto responsable de asesinato el hech
            <dato tipo="val">I</dato>
       </item>
        </item>
      </item>
    </item>
  </item>
```

```
</rep>
```

Un reporte puede chequearse con el programa **sigchq**.

Descripción detallada de cada elemento

`rep` - Raíz de reporte

Puede especificarse el formato de los datos del reporte en el atributo `formato` (ver clasif(5)). Como subelementos del elemento raíz `rep`, pueden mezclarse los elementos `dato` con elementos `item`.

`dato` - Dato.

Permite incluir un dato con un tipo. El tipo se especifica en el atributo `tipo`.

`item` - ítem.

Un ítem del reporte. Un ítem puede constar de datos (elemento `dato`) y subítems (elemento `item`).

Ver también

sigchq(1), reporte(1).

II. Documentación de referencia de programas

sigchq

Nombre

`sigchq` — Chequea un archivo XML para `sigue`.

Synopsis

sigchq [*`opciones`*] *`arch`* [*`arch ...`*]

Descripción

Recibe uno o más archivos XML para `sigue` (DTDs `prcla.dtd`, `individuo.dtd`, `grupo.dtd`, `rep.dtd` o `planest.dtd`) y chequea su sintaxis y convenciones.

Las opciones son las siguientes:

`-ayuda`

: Presenta ayuda corta y opciones

`-D`

: Establece otro directorio en el cual buscar archivos con más precedencia

`-man`

: Presenta página del manual

`-refentry`

: Presenta página del manual en formato DocBook

`-V`

: Versión de este programa

`-f`

: Establece formato de descripciones: `texto`, `tex` o `docbook`

`-l`

: Establece longitud máxima de descripciones. e.g -l 80

`-c`

: Establece nombre del archivo de clasificaciones por defecto (si se omite se usa el mismo nombre del archivo de secuencia/progreso pero con extensión .cla)

`-p`

Indica predicado que debe cumplir el periodo. El predicado debe ser de la forma p=dato o p!=dato o p<dato o p<=dato o p>dato o p>=dato donde cada dato puede ser un flotante, un entero o una cadena (entre apostrofes), pueden usarse paréntesis y los operadores !p, p1 && p2 y p1 || p2. Del predicado se debe deducir un tipo único para p e.g -p "p=’p/2006-1’ || p=’p/2006-2’". Algunos chequeos requieren buscar datos en rangos de periodos, en esos casos se emplea el orden usual del tipo p (lexicográfico en el caso de cadenas)

`-cron`

Especifica archivo con cronograma y tipo de periodos. Extrae periodos válidos del archivo especificado, de las secuencias con el tipo especificado. Separar archivo de tipo con ’:’. e.g -cron "cron.sec:Periodo"

`-g`

Establece predicado para valoraciones de acuerdo al tipo de clasificación. Es de la forma t_1:p_1;t_2:p_2...t_n:p_n donde t_i es un tipo de clasificación referenciada (o vacío y si aplica a valoraciones fuera de clasificación), p_i es un predicado para ese tipo en la variable ’g’ —sintaxis análoga a la de la opción -p— e.g -g ":(g=’D’ || g=’I’) ; Indicador:(g=’E’ || g=’B’ || g=’I’)" Para especificar valoraciones posibles en ejercicio emplee el tipo ‘ejercicio´.

`-n`

No usar archivo de clasificaciones por defecto

`-b`

Indica que no deben reportarse errores cuando falten datos al chequear individuos

`-datos`

Indica datos mínimos que cada persona debe tener y opcionalmente el formato. Separar un tipo de otro con ’;’ y si se desea especificar formato, escribirlo a continuación del tipo separado con ’:’. El formato debe ser de la forma d=dato o d!=dato o d<dato o d<=dato o d>dato o d>=dato donde cada dato puede ser un flotante, un entero o una cadena (entre apostrofes), pueden usarse paréntesis y los operadores !p, p1 & & p2 y p1 || p2. Del predicado se debe deducir un tipo único para d e.g -datos "Nombre:d<>”;Tipo de documento:d=’TI’ || d=’CC’"

`-cursos`

En un plan de estudios, establece orden de cursos. Separar un identificador de otro con ’;’ e.g -cursos "Nivel;Grado"

`-asignaturas`

En un plan de estudios, establece orden de asignaturas. Separar un identificador de otro con ’;’ e.g -asignaturas "Área;Asignatura"

Las convenciones precisas que cada formato debe seguir están en la documentación de cada formato (ver al final de está documentación)

Los errores son reportados por `stderr`. Al completar la revisión si encuentra algún error retornará 1 al sistema operativo (en caso de éxito retorna 0).

El formato de los errores reportados es estándar:

archivo:*linea*:*columna mensaje_error*

y puede ser empleado por varios editores de texto (e.g vi o emacs) para ubicar automáticamente la posición del error.

Ejemplos

```
sigchq bot.prc
```

Revisa el archivo de progreso `bot.prc`

```
sigchq -D /usr/share/xml/structio bot.prc bot.ind
```

Que revisa el archivo de progreso `bot.prc` y el archivo de individuo `bot.ind`, pero emplea con preferencia la ruta `/usr/share/xml/structio` para buscar archivos XML y DTDs.

Ver también

prcla(5), grupo(5), individuo(5), planest(5), rep(5),

reporte

Nombre

`reporte` — Genera reportes con valoraciones de uno o más estudiantes.

Synopsis

reporte [*`opciones`*] [*`dir_est`* | *`grp`*] [[*`dir_est`* | *`grp`*] ...]

Descripción

Recibe uno o más archivos con grupos (ver grupo(5)) o directorios de estudiantes con su progreso. Genera todos los reportes de los individuos del grupo y de los estudiantes en XML o empleando una

plantilla que especifica como debe ser cada reporte. La plantilla que **repasa** puede completar puede ser un documento HTML o un texto, un documento DocBook, LaTeX, RTF o cualquier otro formato que se represente en un texto plano y que sea apropiado para su institución (no es posible emplear una plantilla en un formato binario como el de MS-Word).

Las opciones son las siguientes:

`-ayuda`

Presenta ayuda corta y opciones

`-D`

Establece otro directorio en el cual buscar archivos con más precedencia

`-man`

Presenta página del manual

`-refentry`

Presenta página del manual en formato DocBook

`-V`

Versión de este programa

`-anota`

Especifica una anotación del reporte y su valor. La sintaxis es ’anotación:valor’ Por ejemplo -anota "Fecha:4/4/2012"

`-mder`

Cadena que identifica fin de un macro que debe remplazarse (e.g -mder @)

`-formato`

Formato al que debe convertirse cada dato encontrada en los archivos de entrada. Las posibilidades son texto, texinfo, tex y docbook. Por defecto se usa texto.

`-planest`

Archivo con plan de estudio (niveles, grados, asignaturas, DTD planest.dtd)

`-datosind`

Nombre de archivo con datos de individuo por defecto es ‘datos.ind´.

`-mizq`

Cadena que identifica inicio de un macro por remplazar (e.g -mizq @)

`-anotaniv`

Especifica anotación que debe coincidir con el nivel/grado/etapa y su valor. La sintaxis es ’nivel:valor’ Por ejemplo -anotaniv "Grado:g4"

`-o`

Especifica forma del nombre del archivo de salida de cada individuo al procesar un grupo, puede ser "stdout" para indicar salida estándar o una cadena que puede incluir %d que indica nombre del directorio de los datos del individuo, %c que indica un número consecutivo generado automáticamente. Por defecto es stdout. e.g -o "/tmp/rep5-%c.tex"

`-anotaper`

Especifica anotación que debe coincidir con el periodo y su valor. La sintaxis es ’anotación:valor’ Por ejemplo -anotaper "Periodo:1"

`-texto`

Texto de reporte para un tipo, se especifica como ’tipo :!: encabezado :!: pie’, el encabezado y pie puede tener (dependiendo del tipo) los macros desc, val y otras anotaciones definidos en los archivos de entrada (ver también opciones -i y -d), puede tener también las secuencias de escape \n y \t. Por ejemplo si @ es la cadena que identifica inicio y fin de macro y si se han definido las anotaciones intensidad y Fallas en un tipo Asignatura: -texto "Asignatura: Calificaciones de @desc@. (@intensidad@) F:@Fallas@"

`-prom`

Especifica archivo en el que está el programa para computar promedios. Por ejemplo -prom "mediana.prom"

`-prom_tipos`

Especifica tipos en los que debe agregarse computo de promedios. Por ejemplo -prom_tipos "cadena:Asignatura;Reporte"

`-prom_rango`

Establece periodo inicial y periodo final para el computo de promedios. Separar uno de otro con ’:’. Por ejemplo: -prom_rango "p/2006-1:p/2006-4"

Por cada estudiante, **reporte** lee información de diversas fuentes (línea de comandos, plan de estudio, archivos de clasificaciones, archivos de progreso) que organiza como ítems y datos a partir de los cuales puede calcular promedios y bien producir un archivo XML o bien efectuar remplazos en la plantilla. A continuación se presenta parte de un archivo XML generado:

```
<item tipo="Asignatura">
  <dato tipo="Moderador.Nombres">Pigmaleon</dato>
  <dato tipo="intensidad">1</dato>
  <dato tipo="id">castellano</dato>
  <dato tipo="Fallas">3</dato>
  <dato tipo="computo">Deficiente</dato>
  <item tipo="Logro">
    <dato tipo="id">m1-2001.cla:l1</dato>
    <dato tipo="desc">Logro 1</dato>
    <item tipo="Indicador">
      <dato tipo="id">m1-2001.cla:l1.1</dato>
      <dato tipo="desc">Indicador 1-1 m1</dato>
```

```
    <dato tipo="val">I</dato>
  </item>
```

Note que la información se divide en ítems (de acuerdo al plan de estudios, las clasificaciones y la clasificación elegida para valorar con la opción) y cada ítem tiene datos y posiblemente subítems. En el ejemplo presentado Logro es un subítem de Asignatura, e Indicador es un subítem de Logro.

Para remplazar en una plantilla es necesario que la plantilla se divida en varias partes, una por cada item que reporte emplea (verlos generando primero el XML). Cada una de esas partes puede incluir porciones que se remplazaran con los datos recolectados y que se distinguen con cadenas especiales a izquierda y derecha (ver opciones `-mizq` y `-mder`). Por ejemplo si la información recolectada se ha dividido en ítems y subítems así:

```
Reporte
 |- Nivel
     |- Grado
         |- Área
             |- Asignatura
                 |- Logro
                     |- Indicador
```

y se deseara generar un reporte como texto plano, podría emplearse una plantilla para el item Asignatura como la siguiente (especificada con el parámetro `-texto`):

```
"Asignatura:!:  @id@ (@Fallas@/@intensidad@) Moderador: @Moderador.Nombres@
:!: Promedio: @computo@ "
```

Que indica que se trata de plantilla par asignaturas, un encabezado y un pie para cada asignatura (se separa uno de otro con la cadena ":!:"). Note que se emplea como cadena especial @ tanto a izquierda como a derecha. Si reporte recolecta la información antes presentada en XML, el resultado tras el remplazo del encabezado sería:

```
castellano (3/1) Moderador: Pigmaleon
```

y el pie (después de presentar plantillas de subítems) sería:

```
Promedio: Deficiente
```

Por línea de comandos reporte puede recibir bien uno o más directorios con información de estudiantes y/o uno o más archivos de grupos que referencien datos de estudiantes. Por cada directorio de estudiante que recibe trata de abrir el archivo de individuos `datos.ind` (ver opción `-datosind`) y procesa progreso disponible en el mismo directorio. Por cada archivo de grupo que recibe, realiza el proceso anterior con cada individuo referenciado (suponiendo que el directorio donde está el progreso es el mismo donde está el archivo de individuo).

La información de cada estudiante la agrega al item raíz ‘Reporte´, así como las anotaciones que reporte recibe por línea de comandos: periodo (ver opción `-anotaper`), grado (ver opción `-anotagrado`) y otras (ver opción `-anota`) y las anotaciones que haya en la raíz del plan de estudio. Del plan de estudios agrega ítems por cada elemento curso que preceda o siga al grado especificado en línea de comandos (con la opción `-anotagrado`) y por cada asignatura de ese grado. Por cada asignatura agrega si es el caso información del moderador en datos con identificación de la forma `Moderador.`*`id`* (por ejemplo

`Moderador.Nombres` si `Nombres` fuera un dato del archivo de individuo del moderador), por cada asignatura que tenga asociado un archivo de progreso (con el atributo `prc`) agrega ítems de acuerdo a la información que haya en el archivo de progreso del estudiante para el periodo especificado en línea de comandos (con la opción `-anotaper`). De haberse especificado un programa para realizar promedios (opción `prom`) y tipos de item (opcion `prom_tipos`) en los cuales realizarlo, efectúa promedios en cada item especificado y los agrega como datos de tipo `computo`. El programa en lenguaje `prom` recibirá las variables:

`ejecutor`

con valor `reporte`

`tipo`

con el tipo del item, por ejemplo `Asignatura` si es un item y se especificó con la opción `-prom_tipos`.

`id_item`

Una variable por cada dato del item, el nombre es el identificador y el valor el dato en sí.

`val`

Vector con las valoraciones recolectadas, indexadas en el orden en el que aparecen en los datos y subítems.

`val_tipo`

Vector con los tipos de ítems en los que hay valoraciones (misma numeración de `val`), por ejemplo un valor típico es `Indicador`.

`val_id`

Vector con identificaciones de los ítems en los que hay valoraciones (misma numeración de `val`).

Ejemplos

```
reporte -anotaniv "Grado:g1" -anotaper "Periodo:p2001-1" \
-planest asig.planest -anota "Fecha: 20/4/2002" \
-texto "Reporte:!: Estudiante: @Apellidos@ @Nombres@ Grado: @Grado@
Periodo:@Periodo@  Fecha del boletín: @Fecha@\n:!:Fin del reporte\n" \
-texto "Asignatura:!:   @desc@ (@Fallas@/@intensidad@)\n:!:"  \
-texto "Indicador:!:     @desc@ @val@\n:!:" \
-o "stdout" -mizq @ -mder @ t4.grp
```

Genera reportes por salida estándar de todos los estudiantes del grupo `t4.grp` para el periodo `p2001-1`, suponiendo que el grado es `g1`. El plan de estudios está en el archivo `asig.planest`, la fecha del reporte es `20/4/2002`. La salida generada para cada reporte es un texto plano.

```
reporte -anotaniv "Grado:g1" -anotaper "Periodo:p2001-1" \
```

```
-planest asig.planest -anota "Fecha: 20/4/2002" \
-texto "Reporte:!: Estudiante: @Apellidos@ @Nombres@ Grado: @Grado@
Periodo:@Periodo@  Fecha del boletín: @Fecha@ Comentario: @Comentario@\n
:!:Fin del reporte\n" \
-texto "Grado:!: Grado @desc@  Moderador: @Moderador.t1@\n:!:Fin de Grado" \
-texto "Asignatura:!:@desc@ (@Fallas@/@intensidad@)
Moderador: @Moderador.t2@ Comentario: @Comentario@\n:!:"
-texto "Indicador:!:@desc@ @val@\n Comentario: @Comentario@\n:!:" \
-mizq @ -mder @ estudiante/robcom
```

Genera un reporte a partir del progreso del directorio `estudiante/robcom` del grado `g1` referenciado en el plan de estudios `asig.planest` para el periodo `p2001-1`.

Ver también

rep(5)

iniperiodo

Nombre

`iniperiodo` — Agrega anotaciones y/o valoraciones a archivos de progreso de materias y estudiantes referenciados en un plan de estudios.

Synopsis

iniperiodo [*`opciones`*] *`arch_planest`* ...

Descripción

Recibe uno o más planes de estudio (ver planest(5)), información sobre el periodo (`-p`), el tipo que se valora (`-tipoval`) y bien una valoración por defecto (`-val`) a insertar o información para calcular promedios. Inserta valoraciones pendientes e inserta las valoraciones por defecto o computadas en todos los archivos de progreso de estudiantes referenciados en el plan de estudio en el periodo dado. Opcionalmente puede insertar anotaciones (`-anota`).

Empleando opciones en la línea de comandos puede:

- Evitarse la inserción de valoraciones pendientes (`-nopendientes`) o sólo insertar valoraciones pendientes (`-solopendientes`). Las valoraciones que se consideran pendientes se especifican con la opción `-pendiente`. Pueden marcarse clasificaciones no recuperables con un atributo, que **iniperiodo** puede emplear (`-norec`).

- Limitar los cursos que se examinan (`-curso`) o las asignaturas (`-asignatura`) o los estudiantes (`-estudiante`).
- Pueden calcularse valoraciones de una de las siguientes formas:

  1. A partir de valoraciones existentes. Especificar programa para calcular promedio (`-prom`), periodo inicial, periodo final (`-prom_rango`) y si los periodos no están ordenados lexicográficamente el orden en los periodos (`-ordper`).
  2. A partir de resultados en ejercicios con la opción `-ejercicios`. Especificar programa para calcular promedios (`-prom`), periodo inicial y periodo final (`-prom_rango`), orden en los periodos (`-ordper`), cronograma (`-cron`), tipo que designa periodos en el cronograma (`-tipoper`) y valoración por defecto (`-val`).
  3. Agregar sólo anotaciones (usando sólo las opciones `-anota` y `-nopendientes`).

  Por defecto las valoraciones computadas no remplazan valoraciones existentes, sino que sólo se insertan de no existir. Pueden obligarse el remplazo de valoraciones existentes (`-remplazar`), o para depurar puede inhibirse escribir en los archivos (`-noescribir`).

Las opciones son las siguientes:

`-ayuda`

: Presenta ayuda corta y opciones

`-D`

: Establece otro directorio en el cual buscar archivos con más precedencia

`-man`

: Presenta página del manual

`-refentry`

: Presenta página del manual en formato DocBook

`-V`

: Versión de este programa

`-prom`

: Especifica archivo en el que está el programa para computar promedios. Por ejemplo -prom "mediana.prom"

`-prom_tipos`

: Especifica tipos en los que debe agregarse computo de promedios. Por ejemplo -prom_tipos "cadena:Asignatura;Reporte"

`-prom_rango`

: Establece periodo inicial y periodo final para el computo de promedios. Separar uno de otro con ':'. Por ejemplo: -prom_rango "p/2006-1:p/2006-4"

`-solopendientes`

Sólo inserta calificaciones pendientes.

`-curso`

Fija identificación de cursos que se actualizarán (por defecto se actualizan todos). Separar uno de otro con ’;’. Por ejemplo -curso "g1;g2"

`-asignatura`

Fija identificación de asignaturas que se actualizarán (por defecto se actualizan todas). Separar una de otra con ’;’. Por ejemplo -asignatura "castellano;religion"

`-estudiante`

Fija login de estudiantes que se actualizarán (por defecto se actualizan todos). Separar uno de otro con ’;’. Por ejemplo -estudiante "juaper;romhin"

`-anota`

Establece anotaciones por agregar a la raíz de cada archivo de progreso. Separar tipo de anotación de dato con ’:’ y separar varios pares anotación-valor con ’;’ e.g -a "Fallas:0;Observaciones:". Como atributo valor se empleará el periodo.

`-pendiente`

Establece conceptos que se consideran pendientes/insuficientes. Separar uno de otro con el caracter ’;’ e.g -pendiente "I;D"

`-rempend`

Establece conceptos con los que se remplazan pendientes. De haber, deben ser tantos como conceptos pendientes. Separar uno de otro con el caracter ’;’ e.g -pendiente "I;D" -remped "L;L"

`-tipoper`

Tipo empleado para designar el periodo en archivo de secuencia, por defecto es "Periodo"

`-noescribir`

Indica que no debe escribirse en los .prc, sino que debe reportarse por stderr lo que haría (útil para probar programas prom).

`-nopendientes`

Indica que no deben insertarse valoraciones pendientes de periodos anteriores.

`-ejercicios`

Calcula promedio de ejercicios en una clasificación (las que tengan el tipo especificado con -t)

`-ordenper`

Establece orden en los periodos. Escribirlos en orden ascendente, separados unos de otros con el caracter ’;’ e.g -j "p2001-1;p2004-1;p2002-2". Si no se especifica se usa orden lexicográfico.

`-p`

Periodo (e.g -p "p/2006-2")

`-norec`

Establece tipo para anotaciones y valor de clasificaciones no recuperables (por defecto tipo "recuperable" y valor "no"). Separar una de otra con ;, e.g -norec "recuperable;no".

`-remplazar`

Indica que los promedios que ya existan deben remplazarse con resultados de promediar de nuevo.

`-tipoval`

Tipo que se valora (a las clasificaciones de este tipo del periodo especificado se agregan las valoración por defecto o los promedios de ejercicios, o en caso de calculo de promedios por asignatura son las empleadas en promedios) e.g -tipoval "Indicador".

`-val`

Valoración por defecto para clasificaciones no pendientes o para clasificaciones sin ejercicios (e.g -v "B")

`-cron`

Establece archivo de secuencia con el cronograma de periodos, debe tener tipos ‘Periodo’ con identificadores como los especificados en -prom-rango y fechas de inicio y fin

En los programas para calcular promedios pueden emplearse:

`id`

que debe tener identificación de la asignatura

`tipo`

que es tipo de la asignatura.

`ejecutor`

que debe ser la cadena `iniperiodo`.

`val`

es un vector con todas las valoraciones del año en un orden arbitrario.

`val_tipo`

es un vector con los tipos de las valoraciones de val (val_tipo[0] tiene el tipo de val[0]).

`val_id`

es un vector con las identificaciones de las valoraciones de val (misma convención de val_tipo)

`val_arch`

es un vector con los nombres de los .cla en los que están los indicadores de val (misma convención de val_tipo).

`val_periodo`

es un vector con los periodos de las valoraciones de val (misma convención de val_tipo)

Tenga en cuenta que al calcular promedios de valoraciones sobre un rango de periodo, se acumularán las valoraciones más recientes de cada clasificación (por ejemplo si un indicador tiene una valoración en el periodo 1 y otra en el periodo 2, el programa que calcula promedios sólo recibirá la del periodo 2).

Ejemplos

```
iniperiodo -tipoval "Indicador" -nopendientes -val "B" \
-p "p2001-1" -estudiante t1 insval.planest
```

Inserta la valoración por defecto `B` en los archivos de progreso de todos los estudiantes de grupos referenciados en `insval.planest`, en todos los indicadores referenciados desde secuencias referenciadas en `insval.planest` cuyo periodo sea `p2001-1`.

```
iniperiodo -tipoval "Indicador" -solopendientes \
-norec "Recuperable;no" -pendiente "I" -p "p2001-3" \
-estudiante t1 inspendval.planest
```

Inserta las valoraciones de indicadores pendientes y recuperables hasta el periodo `p2001-3` en los archivos de progreso del estudiante con login `t1` que estén referenciados en `inspendval.planest`. Considera que la única valoración pendiente es `I` y que los indicadores no recuperables tienen una anotación con tipo `Recuperable` y como dato `no`.

```
iniperiodo -prom t2.prom -prom_rango "p2001-1:p2001-2" \
-tipoval "Indicador" -norec "Recuperable;no" -pendiente "I" \
-p "p2001-3" -estudiante t1 inspendval.planest
```

La misma funcionalidad del ejemplo anterior, pero además calcula promedios de indicadores entre los periodos `p2001-1` y `p2001-2` usando el script `t2.prom` y agregando el promedio calculado en la raíz del archivo de progreso con periodo `p2001-3`

Ver también

progcla(5), clasificacion(5)

Fallas

No maneja entidades externas. Es decir si un archivo de progreso emplea entidades externas, al actualizarlo, lo escribirá incluyendo la entidad externa, eliminando la declaración e incluyendo el DTD como DTD interno.

camval

Nombre

`camval` — Cambia valoraciones en un archivo de progreso en clasificaciones

Synopsis

camval [*`opciones`*] *`arch_prc`* [*`archcla perval`*]+

Descripción

Recibe el archivo de progreso por modificar (ver progcla(5)) y cuatruplas que especifican las valoraciones por modificar del archivo de progreso. Cada cuatrupla consta de (en este orden): archivo de clasificación, identificación de la clasificación, periodo y valoración.

Las opciones son las siguientes:

`-ayuda`

: Presenta ayuda corta y opciones

`-D`

: Establece otro directorio en el cual buscar archivos con más precedencia

`-man`

: Presenta página del manual

`-refentry`

: Presenta página del manual en formato DocBook

`-V`

: Versión de este programa

Si durante el procesamiento se encuentra algún error (por ejemplo una clasificación especificada en la línea de comandos que no esté referenciada en el archivo de progreso), no modifica el archivo de progreso.

Ejemplos

```
camval t2.prc "m1-2001.cla" "l1.1" "p2001-2" "D"
```

En el archivo `t2.prc` modifica la valoración que indica progreso de la clasificación `l1.1` del archivo de clasificaciones `m1-2001.cla` en el periodo `p2001-2`, la nueva valoración será `D`.

Ver también

progcla(5), clasificacion(5)

camanota

Nombre

`camanota` — Cambia datos de anotaciones en un archivo de progreso en clasificaciones

Synopsis

camanota [*`opciones`*] *`arch_prc`* [*`typ per val`*]+

Descripción

Recibe el archivo de progreso por modificar (ver progcla(5)) y triplas de la forma (tipo,periodo,valor), que indican el valor que deben tener anotaciones con el tipo y el periodo dado.

Las opciones son las siguientes:

`-ayuda`

: Presenta ayuda corta y opciones

`-D`

: Establece otro directorio en el cual buscar archivos con más precedencia

`-man`

Presenta página del manual

`-refentry`

Presenta página del manual en formato DocBook

`-V`

Versión de este programa

Ejemplos

```
camanota m1.prc Fallas p/2006-2 10
```

En el archivo `m1.prc` inserta una anotación para el periodo `p/2006-2` con tipo `Fallas` y con el dato `10`.

Ver también

progcla(5),

Bibliografía

Aprendiendo a aprender Linux: Guías para colegios con plataforma de referencia S-Helio 1.1.

Manual de uso de `repasa`.

Archivo de legislación y jurisprudencia sobre educación y propiedad intelectual en Colombia.

Ley 115 de 1994. Ley General de Educación en Colombia
http://structio.sourceforge.net/leg/1994ley115.html.

Decreto 1860 de 1994 http://structio.sourceforge.net/leg/1994dec1860.html.

Tercer curso gratuito de Linux en colegios de Colombia.

Índice