Capítulo 1

Conceptos importantes

1.1 Reglas

Una *regla* es una descripción formal de cómo arara maneja una cierta tarea. Por ejemplo, si queremos usar `pdflatex` con nuestra herramienta. Debemos tener una regla para ello. A las directivas se le asignan reglas, así si llamamos a una regla `foo` no existente, por ejemplo, de hecho provocará un error:

Cuando una regla es definida, arara proporciona automáticamente una capa de acceso a esa regla a través de directivas en el código fuente, un concepto para ser introducido formalmente más adelante, en la sección 2.2. Observe que una directiva refleja una instancia particular de una regla del mismo nombre (es decir, una directiva `foo` en un código fuente determinado es una instancia de la regla.

En resumen, una regla es un archivo de texto plano escrito en el formato YAML, descrito en el capítulo 10, en la página 150. Opté por este formato porque en ese entonces era más limpio y más intuitivo de usar que otros lenguajes de marcado como XML, además de ser un estándar de serialización de datos para lenguajes de programación.

Las reglas predeterminadas, es decir, las reglas enviadas con arara, se colocan dentro de un subdirectorio especial llamado `rules/` dentro de otro directorio especial llamado `ARARA_HOME` (el lugar donde nuestra herramienta esta instalado). Más adelante, en la sección 4.2, en la página 43, aprenderemos que podemos agregar un número arbitrario de rutas para almacenar nuestras propias reglas, en orden de prioridad, por lo que no se preocupe demasiado por la ubicación de las reglas predeterminadas, aunque es importante comprender y reconocer su existencia.

La siguiente lista describe la estructura básica de una regla arara presentando los elementos apropiados (o claves, si consideramos la nomenclatura YAML adecuada). Observe que los elementos marcados como **M** son obligatorios (es decir, la regla *tiene* que estar para que funcionen). De manera similar, los elementos marcados como **O** son opcionales, por lo que puede ignorarlos de manera segura al escribir una regla para nuestra herramienta. Una clave prece-

dida por `context`→ indica un contexto y debe definirse adecuadamente dentro de ella.

1. `!config` Esta palabra clave es obligatoria y debe ser la primera línea de cualquier regla arara. Denota los metadatos de mapeo de objetos para ser utilizados internamente por la herramienta. En realidad, la herramienta no es demasiado exigente para usarla (de hecho, podría suprimirla por completo y arara no se quejará), pero se considera una buena práctica comenzar con todas las reglas con una palabra clave `! Config` independientemente.

2. `identifier` Esta clave actúa como un identificador único para la regla (como se esperaba). Se recomienda utilizar letras minúsculas sin espacios, acentos o símbolos de puntuación, como buena práctica (otra vez). Como convención, si tiene un identificador llamado `pdflatex`, el nombre de archivo de la regla debe ser `pdflatex.yaml` (como nuestra propia instancia). Tenga en cuenta que, aunque también se sabe que `yml` es una extensión YAML válida, arara solo considera los archivos que terminan con la extensión `yaml`. Esta es una decisión deliberada.

3. `name` Esta clave contiene el nombre de la *tarea* (una regla instanciada a través de una directiva) como una cadena simple. Cuando se ejecuta arara, este valor se mostrará en la salida entre paréntesis.

4. `authors` Nos encanta culpar a la gente, así que arara presenta una clave especial para nombrar a los autores de la regla (si corresponde) para que pueda escribirles comunicaciones electrónicas severas. Esta tecla contiene una lista de cadenas. Si la regla solo tiene un autor, agréguela como el primer (y único) elemento de la lista.

5. `comands` Esta clave se introduce en la versión 4.0 de arara y denota una posible lista de comandos. Desde la perspectiva del usuario, cada comando se denomina *subtarea* dentro del contexto de una tarea (regla y directiva). Una tarea puede representar un solo comando (una subtarea única), así como una secuencia de comandos (subtareas). Por ejemplo, la regla `frontespizio` requiere al menos dos comandos. Entonces, como un medio para normalizar la representación, una tarea compuesta de un solo comando (subtarea única) se define como el único elemento de la lista, en oposición a las versiones anteriores de arara, que tenía una clave específica para mantener solo un comando.

Incompatibiliadad con veriones anteriores

Estimado lector, tenga en cuenta que las reglas de la versión 4.0 son incompatibles con versiones anteriores de arara. Si está migrando de versiones anteriores a la versión 4.0, debemos reemplazar `comando` por `comandos` y especificar un elemento contextual, como se ve en las siguientes descripciones.

Consulte la Sección 2.3, en la página 24, para obtener una guía de migración comprensible.