**Rúbrica 4: Versionamiento del software**

En esta rúbrica vamos a tratar las maneras en las que se liberan las distintas versiones del software y cómo suelen denominarse distintos aspectos de este ámbito.

Una **versión de software** es una identificación única que sirve para identificar una pieza concreta de software en un estado determinado.

Hay varias formas de nombrarlas, desde una ordenación secuencial (1.0 -> 1.1 -> 1.2 ...), usando nombres y fechas (las versiones de Ubuntu por ejemplo, se numeran con los últimos dígitos del año y el mes de salida, que puede ser abril u octubre. Además se nombra con un animal y adjetivo que empiezen por la misma letra. Por ejemplo: Ubuntu Hardy Heron / Ubuntu 8.04) o hasta añadiendo dígitos a 3 para ser una aproximación a pi (por ejemplo, la versión actual de TeX es 3.14159265).

Estas formas son muy variables, así que vamos a tomar como referencia principal el sistema de numeración que usa el kernel de Linux actualmente.

Este sistema es un método *secuencial*, que permite identificar al software por un conjunto de tres dígitos, separados por puntos:

* El primer dígito hace referencia a la versión ***major***
* El segundo dígito hace referencia a la versión ***minor***
* El tercer dígito hace referencia a las versiones ***patch***
* A esto se le suelene añadir una serie de "etiquetas" que explicaremos más adelante

En la práctica pueden añadirse números adicionales. Por ejemplo, Linux suele incorporar un cuarto número, que refleja cambios tan pequeños que no podrían justificar un nuevo número, pero que son de vital importancia, o bien para agrupar paquetes de arreglo en un número más compacto. Estas revisiones se usan por ejemplo en los sistemas operativos. Cuando Ubuntu ha sufrido varios procesos de actualización, dentro de la misma versión, se sube una iso con este número aumentado (16.04 -> 16.04.1) que incorpora estos arreglos y actualizaciones, para que no tengan que ser descargados de nuevo.

Las versiones ***major*** implican grandes cambios a nivel interno, generando incompatibilidades (o al menos no pudiéndose garantizar la compatibilidad) con versiones anteriores del software. En el caso de Linux, también se produce un cambio en este número por otras razones. Por ejemplo, el paso de 2.x a 3.x sucedió porque los demás números eran muy grandes y se decidió aumentarlo para además darle énfasis al 20 aniversario del kernel.

Ejemplos de estos cambios sería eliminar funciones que ya no se usen, cambiar o eliminar constantes...

Las versiones **minor** reflejan cambios significativos al software que, sin romper compatibilidades, agregan nuevas funciones.

Ejemplos de estos cambios serían agregar, depreciar o modificar funciones (recordando que esta función debe seguir declarada para poder compilar sin problemas) o hacer lo propio con constantes.

Las versiones ***patch*** consisten en bugfixes y arreglos de los fallos observados.

Estos cambios están limitados a modificar las funciones implementadas para evitar errores.

Analizando los [changelog](https://kernelnewbies.org/LinuxVersions) del Kernel, podemos ver qué cambios han ido sucediéndiose en cada versión (sin contar los bugfixes). Aquí puede comprobarse que los cambios 2 -> 3 y 3 -> 4 tienen poca justificación técnica y están más fundamentados en razones estéticas y de comodidad.

Adicionalmente, se ha introducido la idea del uso de modificadores para complementar a la versión. Esto radica de la necesidad de indicar aspectos inherentes a la versión, como si es experimental, si puede usarse en ambientes de producción, cuantos cambios ha sufrido, etc...

Volviendo a tomar como estándar el kernel de Linux, anteriormente esto se reflejaba en la propia numeración. Números impares (2.3) hacían referencia a versiones de desarrollo, mientras que los pares, a estables (2.4). Esto se abandonó a partir de las versiones 2.6.

Algunos de estos términos generales son:

* **LTS**: Long term support. Es una versión que recibirá soporte pronlongado (es muy común en los sistemas operativos, que reciben actualizaciones por X tiempo. Estas versiones LTS marcan mayores tiempos en los que se recibe este soporte.
* **Pre-alpha**: es una versión que, a pesar de ser lanzada, ni está completa ni representa cómo será el producto final. Es una pieza de software que aún está sujeta a cambios en funcionalidad y enfoque.
* **Alpha**: es una versión que, aunque en principio está terminada en el ámbito de diseño (es decir, no se planea incorporar funcionalidad adicional), su código no es definitivo porque, o bien funciona tan sólo en parte, o bien tiene fallos y el software es inestable. Se suele representar como XX.XX.XXa.
* **Beta:** es una evolución de la versión alfa en la que, a pesar de presentar aún errores, es buen reflejo de lo que será el producto final. El software no es estable, pero los errores suelen ser menos frecuentes y graves. Se suele representar como XX.XX.XXb.
* **RC:** Release candidate (que al español vendría a traducirse como *candidata al lanzamiento*) es una versión que ya ha sido probada y controlada. Es el producto resultante del arreglo de los fallos de las versiones anteriores y es la que finalmente tendrá la salida destinada al usuario final, a no ser que se detecte algún fallo grave, que antes hubiera pasado desapercibido, que requiera arreglo. Se suele representar como XX.XX.XX-RCn (siendo n el número de RC, que puede haber varios).
* **Milestone:** La traducción al español es hito, por lo que hace refernecia a una versión de software que ha cumplido una serie de los objetivos fijados cuando se planificó el proyecto. Se suele representar como XX.XX.XX-Mn.
* **Snapshot:** Traducido como "captura", es un concepto similar a las *daily builds* o *nightly.* Son versiones que incorporan los cambios más inmediatos en un software que está sufriendo un proceso activo de desarrollo.

**Referencias**

http://semver.org/

http://apr.apache.org/versioning.html

http://stackoverflow.com/questions/4277297/what-is-snapshot-builds-sources-version

wikipedia.org

Especialmente:

https://en.wikipedia.org/wiki/Linux\_kernel#Version\_numbering

https://en.wikipedia.org/wiki/Software\_release\_life\_cycle

https://en.wikipedia.org/wiki/Software\_versioning

http://www.proyectoautodidacta.com/conociendo/los-estados-y-las-versiones-del-software/

http://www.cristalab.com/blog/control-de-versiones-de-software-c82353l/