Algunas recomendaciones para UNIX/LINUX y Windows

Recomendamos realizar este curso en el S.O UNIX/LINUX, en Linux o en MAC (traen UNIX de serie). Esta mini-guía contiene un resumen de UNIX y sus equivalente Window. Usar Windows da problemas. Este documento ayudar a solucionarlos o si no buscando la solución en Internet.

Uso del terminal de comandos en UNIX/LINUX

Los comandos de Git, node o npm se deben ejecutar en un terminal de comandos. Un programa denominado shell atiende el terminal de comandos. Este suele indicar que esta preparado con: **$**. Un comando se ejecuta tecleando su nombre y parámetros, seguidos de retorno de línea.

En UNIX/LINUX el terminal de comandos suele tener un icono asociado en algún lugar del escritorio. El terminal de comandos se abre al hacer clic en el icono.

Un terminal de comandos tiene siempre un **directorio de trabajo** (working directory o **wd**) asociado, que es el directorio sobre el cual se ejecutan los comandos. Por ejemplo

```
$ ls                    ## lista ficheros del wd (directorio de trabajo)
$ ls dir1               ## lista ficheros del directorio dir1, contenido en wd
$ ls ../dir2            ## lista ficheros del directorio dir2, contenido en el directorio padre de wd
$ ls /home/eva/dir3     ## lista ficheros de identificado por la ruta absoluta dir3

$ nano fichero.ext      ## edita, si existe, o crea, si no existe, el fichero.js del directorio de trabajo
$ vi fichero.ext        ## edita, si existe, o crea, si no existe, el fichero.js del directorio de trabajo

$ pwd       ## muestra la ruta absoluta al directorio de trabajo del terminal de comandos

$ cd dir1   ## cambia el directorio de trabajo por el directorio dir1 contenido en él
$ cd ..     ## cambia el directorio de trabajo por su directorio padre (el primero en la ruta a la raíz)
```

El sistema de ficheros y directorios de UNIX/LINUX tiene estructura de árbol y los ficheros se identifican en los comandos con rutas absolutas, que comienzan en el directorio raíz del árbol (identificado con /), o relativas, que comienzan en el wd (directorio de trabajo). Las rutas son caminos en el árbol de directorios. Por ejemplo **dir1** y **../dir2** son rutas relativas a wd, mientras que **/home/eva/dir3** es una ruta absoluta que parte del directorio raíz que identifica el directorio **dir3**, contenido en el directorio **eva**. **eva** a su vez está contenido en el directorio **home**, que está contenido en el directorio raíz **/**.

Los ficheros y directorios de un ordenador pueden verse y modificarse de 2 formas:

1. A través del terminal de comandos invocando comandos.
2. A través de ventanas con iconos que muestran gráficamente el contenido de un directorio. Los comandos se ejecutan al hacer clic con el ratón en el icono.

Ambas formas de ver y procesar ficheros son equivalentes y muestran la misma información. Por ejemplo: El comando **ls** muestra los mismos ficheros que la ventana con iconos asociada al mismo directorio. O si borramos o modificamos el nombre de un fichero con comandos del terminal de comandos, la ventana de iconos quitará el icono o mostrará el nuevo nombre.

Listado de Comandos UNIX/LINUX mas habituales:
http://www.cheat-sheets.org/saved-copy/fwunixref.pdf
Pequeño tutorial de de UNIX/LINUX:
http://www.cheat-sheets.org/saved-copy/A.very.brief.guide.to.Linux.htm

Edición de programas en UNIX/LINUX

Los ficheros que contienen programas deben editarse con editores de texto plano, tales como nano, vi o vim, notepad o notepad++, sublime text, ATOM, Brackets, … Estos editores solo incluyen los caracteres que muestran y permiten que los programas editados con ellos sean ejecutados sin problemas.

¡Cuidado! Los editores de documentos como **Word** o **Pages** estropean el texto de un programa porque añaden caracteres especiales que impedirán que se pueda ejecutar. No utilizarlos nunca para editar programas ejecutables, porque los dejan inservibles.

Si ya está utilizando algún editor o IDE que permite editar texto plano y lo conoce bien, lo más recomendable es que lo siga utilizando. Sino le recomendamos uno de estos:

Sublime-text: editor de tipo wysiwyg potente y fácil de usar, existente para UNIX, Windows, ... Se puede descargar una versión de prueba para uso ilimitado de https://www.sublimetext.com.

ATOM: editor de tipo wysiwyg potente y fácil de usar, existente para UNIX, Windows, ... Tiene funciones de IDE y permite inspeccionar y gestionar repositorios Git. Es gratuito (patrocinado por GitHub) y se descarga de https://atom.io/.

nano: editor muy sencillo, auto-explicativo y fácil de aprender, aunque muy limitado. Está en todos los UNIX o Linux. Se invoca en modo comando:

```
$ nano fichero.ext      ## abre para edición, si existe, o crea, si no existe, el fichero.ext en wd
```

vi o vim: vi es el editor tradicional existente de UNIX y está en todos los UNIX o Linux. Es el editor que suele abrir Git por defecto. Es potente, pero orientado a comando y requiere aprender los comandos. vim es una evolución de vi. Se invoca en modo comando:

```
$ vi fichero.ext        ## abre para edición, si existe, o crea, si no existe, el fichero.ext en wd
```

Lista de comandos:
http://www.atmos.albany.edu/daes/atmclasses/atm350/vi_cheat_sheet.pdf

Además están los editores de los IDEs (Integrated Development Enviroments) mas profesionales. Son muy convenientes, porque permiten tanto editar programas, como realizar muchas tareas de gestión de un proyecto software. Destacan **Visual Studio** (https://visualstudio.microsoft.com/es/), que es gratuito, o **Webstorm** (https://www.jetbrains.com/webstorm/), que es de pago pero da licencias gratuitas para actividades educativas de profesores y estudiantes (https://www.jetbrains.com/student/).

Uso del terminal de comandos en Windows

Los comandos de Git, node o npm se deben ejecutar en un terminal de comandos. El terminal de comandos está atendido por un programa, que suele indicar que esta preparado mostrando: **>**. Un comando se ejecuta tecleando su nombre y parámetros asociados, seguidos de retorno de línea.

En Windows hay que hacer clic en el icono de Windows y teclear “powershell” seguido de retorno de línea para abrir el terminal de comandos. Alternativamente se puede utilizar “cmd” (terminal por defecto anterior) o “git bash” (terminal que nos provee Git cuando lo instalamos en Windows).

Un terminal de comandos tiene siempre un directorio de trabajo (working directory o wd) asociado, que es el directorio sobre el cual se ejecutan los comandos. Por ejemplo en Windows

```
> dir                  ## lista ficheros del wd. ls funciona también en “powershell”, pero no en “cmd”
> dir dir1             ## lista ficheros del directorio dir1, contenido en wd
> dir ..\dir2          ## lista ficheros del directorio dir2, contenido en el directorio padre de wd
> dir /home/eva/dir3 ## lista ficheros de identificado por la ruta absoluta dir3

> pwd       ## muestra la ruta absoluta al directorio de trabajo del terminal de comandos

> cd dir1   ## cambia el directorio de trabajo por el directorio dir1 contenido en él
> cd ..     ## cambia el directorio de trabajo por su directorio padre (el primero en la ruta a la raíz)
```

Las últimas versiones de Windows suelen aceptar también comandos de UNIX.

El sistema de ficheros y directorios de Windows tiene estructura de árbol y los ficheros se identifican en los comandos con rutas absolutas, que comienzan en el directorio raíz del árbol (identificado con **C:\ o **), o relativas, que comienzan en el wd (directorio de trabajo). Las rutas son caminos en el árbol de directorios. Por ejemplo **dir1** y **..\dir2** son rutas relativas a wd, mientras que **C:\Users\eva\dir3** o **\Users\eva\dir3** es una ruta absoluta que parte del directorio raíz que identifica el directorio **dir3**, contenido en el directorio **eva**. **eva** a su vez está contenido en el directorio **home**, que está contenido en el directorio raíz **C:** o ****.

Los ficheros y directorios de un ordenador pueden verse y modificarse de 2 formas:

1. A través del terminal de comandos invocando comandos.
2. A través de ventanas con iconos que muestran gráficamente el contenido de un directorio. Los comandos se ejecutan al hacer clic con el ratón en el icono.

Ambas formas de ver y procesar ficheros son equivalentes y muestran la misma información. Por ejemplo: El comando **dir** muestra los mismos ficheros que la ventana con iconos asociada al mismo directorio. O si borramos o modificamos el nombre de un fichero con comandos del terminal de comandos, la ventana de iconos quitará el icono o mostrará el nuevo nombre.

Listado de comandos Windows mas habituales:
http://simplyadvanced.net/blog/cheat-sheet-for-windows-command-prompt/

Edición de programas en Windows

Los ficheros que contienen programas deben editarse con editores de texto plano, tales como nano, vi o vim, notepad o notepad++, sublime text, ATOM, Brackets, … Estos editores solo incluyen los caracteres que muestran y permiten que los programas editados con ellos sean ejecutados sin problemas.

¡Cuidado! Los editores de documentos como **Word** estropean el texto de un programa porque añaden caracteres especiales que impedirán que se pueda ejecutar. No utilizarlos nunca para editar programas ejecutables, porque los dejan inservibles.

Si ya está utilizando algún editor o IDE que permite editar texto plano y lo conoce bien, lo más recomendable es que lo siga utilizando. Sino le recomendamos uno de estos:

Sublime-text: editor de tipo wysiwyg potente y fácil de usar, existente para UNIX, Windows, ... Se puede descargar una versión de prueba para uso ilimitado de https://www.sublimetext.com.

ATOM: editor de tipo wysiwyg potente y fácil de usar, existente para UNIX, Windows, ... Tiene funciones de IDE y permite inspeccionar y gestionar repositorios Git. Es gratuito (patrocinado por GitHub) y se descarga de https://atom.io/.

Notepad: es el editor wysiwyg de texto plano de windows. Es recomendable instalar **Notepad++** que soporta análisis sintáctico del lenguaje de programación y se puede descargar de https://notepad-plus-plus.org/.

vi o vim: vi es el editor tradicional existente de UNIX y en Windows se instala cuando se instala Git, pero está accesible solo en el terminal de comandos “git bash”. Además es el editor que suele abrir Git por defecto. Es potente, pero orientado a comando y requiere aprender los comandos. vim es una evolución de vi. Se invoca en modo comando:

```
$ vi fichero.ext        ## abre para edición, si existe, o crea, si no existe, el fichero.ext en wd
```

Lista de comandos:
http://www.atmos.albany.edu/daes/atmclasses/atm350/vi_cheat_sheet.pdf

Además están los editores de los IDEs (Integrated Development Enviroments) mas profesionales. Son muy convenientes, porque permiten tanto editar programas, como realizar muchas tareas de gestión de un proyecto software. Destacan **Visual Studio** (https://visualstudio.microsoft.com/es/), que es gratuito, o **Webstorm** (https://www.jetbrains.com/webstorm/), que es de pago pero da licencias gratuitas para actividades educativas de profesores y estudiantes (https://www.jetbrains.com/student/).

Programas node.js en Windows

node.js se diseño para UNIX y luego se porto a Windows. Hacer programas que funcionen en ambos S.O. requiere conocer las particularidades de Windows. Las soluciones a las entregas de este curso, así como los entornos de test se han probado en Windows para que funcionen correctamente.

A continuación enumeramos aspectos que se deben tener en cuenta para que un programa node.js funcione en Windows:

1) **Caracteres**: Windows no suele tener configurado el código UNICODE como código de caracteres del S.O. En ese caso la consola no mostrará correctamente caracteres no soportados por el código del S.O. como por ejemplo ideogramas chinos. El código enviado por el programa node será el código correcto en UNICODE, pero el S.O. no sabrá como mostrarlo. El programa de pruebas esta en node.js y por lo tanto si detectará que se envían los caracteres correctos.
2) **RUTAS**: La primera diferencia está en las rutas del sistema de ficheros de UNIX, que utilizan **/** para separar directorios, mientras que las de Windows utilizan \. Esto hace que la gestión de rutas en programas node.js no debe utilizar strings como “dir1/dir2/fich1.js”, que solo funcionan en UNIX. Por ello, hay que utilizar el **módulo path** de node.js (https://nodejs.org/en/docs/) para crear rutas a partir de nombres de directorios. La documentación del módulo path está en: https://nodejs.org/dist/latest-v10.x/docs/api/path.html

   Por ejemplo, para importar el módulo contenido en el fichero de nombre mod1.js en el directorio donde está el programa, el siguiente código solo funciona en UNIX:

   ```
   var mod1 = require( “./mod1.js”  );
   ```

   En cambio debemos utilizar el siguiente código:

   ```
   var path = require(‘path’);    // Importar módulo path para construir la ruta con “resolve(..)”
                                  // __dirname indica en node.js la ruta absoluta al programa
   var mod1 = require( path.resolve( __dirname, “mod1.js" ) );
   ```

3) **Enlace simbólicos**: los enlaces simbólicos a ficheros también se gestionan de forma diferente, pero como no se usan en este curso no vamos a entrar en los detalles.