* # Hoja de trucos terminal
* Pwd # imprimir directorio de trabajo
* Is # lista de filas en el directorio
* Cd # cambiar directorio
* ~ # directorio de inicio
* .. # subir un directorio
* -# directorio de trabajo anterior
* Ayuda # obtener ayuda
* -h # obtener ayuda
* --ayuda # obtener ayuda
* Hombre # manual
* Cat # genera el contenido de un archivo
* Mkdir # crear nuevo directorio
* Open # abre un archivo con el programa asociado, un directorio con Finder o una URL con el navegador web predeterminado
* Ps # enumera todos los procesos en ejecución
* Kill # terminar el proceso anterior
* Rmd # eliminar archivo permanente
* Rmdir # eliminar directorio
* ## trabajando con gift
* ### Inicio rápido
* git clone <url> # Clonar directorio
* git checkout -b <nueva-rama> # Crear nueva rama local
* git push -u origin <nueva-rama> # Sincronizar rama local con remota
* git checkout <rama> # Rama de pago
* git push origin <branch> # Empujar rama al control remoto
* git rama -d <nombre de rama> # elimina la rama local
* git push origin:<branchname> # elimina la rama remota
* git subtree push --prefix docs origin gh-pages # empuja documentos como subárbol a gh-pages
* ### Clonar directorio
* git clon <url>
* ### Crear proyecto
* proyecto de cd/
* git init # inicializa el repositorio
* git agregar. # agregar esos archivos 'desconocidos'
* git commit # confirma todos los cambios, edita la entrada del registro de cambios
* git rm --cached <archivo>... # comando ridículamente complicado para deshacer, en caso de que hayas olvidado .gitignore
* ### Ramificación y fusión
* git branch # muestra la lista de todas las ramas (\* está activo)
* git checkout -b linux-work # crea una nueva rama llamada "linux-work"
* <realizar cambios>
* git confirmar -a
* git checkout master # volver a la rama master
* git merge linux-work # fusionar conjuntos de cambios de linux-work (Git >= 1.5)
* git tirar. linux-work # fusionar conjuntos de cambios de linux-work (todas las versiones de Git)
* git rama -m <nombre antiguo> <nombre nuevo> # cambiar nombre de rama
* git rama -m <nuevo nombre> # cambiar el nombre de la rama actual
* ### Eliminar proyecto
* git rama -d <nombre de rama> # elimina la rama local
* git push origin:<branchname> # elimina la rama remota
* git remoto podar <nombre de rama> # actualizar sincronización local/remota
* ### Fusión ascendente
* git remoto -v # Obtener lista de sucursales remotas
* git remoto agregar upstream <upstream github url> # Agregar original como upstream
* git remoto -v # Comprobar aguas arriba
* git fetch upstream # Obtener repositorio original
* desarrollo de git checkout # Cambiar a la rama principal en la bifurcación local
* git merge upstream/development # Fusionar original con fork
* git diff --solo nombre | único | xargs subl # Solucionar conflictos en Sublime Text
* ### Importación de parches
* git aplicar < ../p/foo.patch
* git confirmar -a
* ### Exportación de parches
* <realizar cambios>
* git commit -a -m "mensaje de confirmación"
* git format-patch HEAD^ # crea 0001-commit-message.txt
* # (HEAD^ significa cada parche desde una revisión antes de la
* # punta de la rama, también conocida como CABEZA)
* ### Inspeccionar revisione
* # inspeccionar el historial visualmente
* gitk # esto abre una ventana de Tk y le muestra cómo están conectadas las revisiones
* # inspeccionar el historial
* git log # esto canaliza un registro de la rama actual a su PAGER
* git log -p # lo mismo, pero agrega un parche después de cada mensaje de confirmación
* # inspeccionar un compromiso específico
* git show HEAD # muestra información de confirmación, diffstat y parche
* # de la punta de la rama actual
* ### Refiriéndose a las revisione
* # por nombre
* git log v1.0.0 # muestra el historial hasta la etiqueta "v1.0.0"
* git log master # muestra el historial de la rama "master"
* # relativo a un nombre
* git show master^ # mostrar padre a la última revisión del maestro
* git show master~2 # mostrar al abuelo a la punta del maestro
* git show master~3 # muestra al bisabuelo a la punta del maestro (entiendes la idea)
* # por salida de "git describe"
* git show v1.4.4-g730996f # obtienes esta cadena llamando a "git describe"
* # por hash (internamente, todos los objetos se identifican mediante un hash)
* git mostrar f665776185ad074b236c00751d666da7d1977dbe
* git show f665776 # un prefijo único es suficiente
* # etiquetar una revisión
* git tag v1.0.0 # hacer que el HEAD actual se conozca como "v1.0.0"
* git tag interesante v1.4.4-g730996f # etiqueta una revisión específica (no HEAD)
* ### Comparación de revisiones
* # diferencia entre dos ramas
* git diff origin..master # canaliza una diferencia en PAGER
* git diff origin..master > my.patch # canaliza una diferencia en my.patch
* # obtener diferenciación de trabajo no comprometido
* git diff --stat CABEZA
* ## Sublime como editor de texto predeterminado
* discos compactos ~
* contenedor mkdir
* ln -s "/Aplicaciones/Sublime Text 2.app/Contents/SharedSupport/bin/subl" ~/bin/subl
* git config --global core.editor "subl -n -w
* ### Si eso no funciona
* sudo rm -rf /usr/local/bin/subl
* sudo ln -s /Aplicaciones/Sublime\ Texto\ 2.app/Contents/SharedSupport/bin/subl /usr/local/bin