A collection of all kinds of shell scripts

Hull

Organisation: Copyright (C) 2008-2025 Olivier Boudeville

Contact: about (dash) hull (at) esperide (dot) com

Creation date: Sunday, August 17, 2008

Lastly updated: Friday, February 28, 2025

Version: 1.0.10

Status: Stable

Dedication: Users and maintainers of `Ceylan-Hull`.

Abstract: The role of `Hull` is to concentrate various, generic-purpose convenience shell scripts, on behalf of the Ceylan project.

The latest version of this documentation is to be found at the official Ceylan-Hull website (`http://hull.esperide.org`).

The documentation is also mirrored here.

Overview

Here are **a few scripts, sorted by themes, that may be convenient for at least some users**.

Many of them can display their intended usage on the console by specifying them the `-h` / `--help` command-line option.

In each category, scripts are roughly sorted by decreasing interest/update status. The scripts that we deem the most useful are described more precisely at the end of this document.

As much as possible, we try to name these scripts functionally rather than in terms of tools, so that the implementation they rely upon can be updated as transparently as possible (i.e. with as little change as possible in the user's habits).

These scripts are intended for all kinds of Unices, especially GNU/Linux, yet most of them are believed to work on Windows as well, provided a relevant context exists there.

Each script entry is a link pointing directly to the script itself. They may be directly obtained as a whole from the Hull repository.

Most of them are certainly not rocket science.

Note

A few of these scripts may be a bit outdated, as not all of them are daily used; rely on them with (a bit of) caution! (and tell us if you would like some of them to be updated).

Table of Contents

Listing of the recommended shell scripts per usage

Related to Filesystems

To search for files and content

- wh ("where"): a more convenient `find`; see the wh full usage
- locate.sh: locates the files whose name corresponds to the specified pattern, among the filesystem elements that are routinely scanned
- regrep: recursive grep for the ones that lack it; see the regrep full usage

To list filesystem elements

- list-filesystem-entries-by-size.sh: lists, from any specified directory otherwise from the current one, the direct filesystem entries (local files and directories), sorted by decreasing size of their content
- list-files-in-tree-by-size.sh: lists, from any specified directory otherwise from the current one, all files in tree, sorted by decreasing size of their content
- list-files-in-tree-by-most-recent-modification-time.sh: lists, from any specified directory otherwise from the current one, all files in tree, sorted from the most recently modified to the least
- list-local-directories.sh:: lists all local, direct directories of any specified directory, or from the current one. Useful for overcrowded base directories

To fix names, paths, permissions, content

- fix-filename.sh: "corrects" the name of the specified file (or directory), to remove spaces and quotes (replaced by '-'), accentuated characters in it, etc.
- fix-paths-in-tree.sh: does the same as `fix-filename.sh`, yet in a tree
- fix-file-permissions.sh: corrects, for all files in the current directory, the UNIX permissions for the most common file extensions
- fix-www-metadata.sh: fixes the permissions in the current tree so it can be transferred as a whole without permission errors to a given server more than once (typically useful for third-party elements that are to populate web roots, like Mathjax)
- set-files-unexecutable-in-tree.sh: ensures that all files found recursively from the current directory are not executable
- fix-unbreakable-spaces.sh: removes any unbreakable space in specified text file
- fix-unbreakable-spaces-in-source-tree.sh: removes any unbreakable space in specified tree

- fix-null-width-spaces.sh: removes any null-width (invisible Unicode) space into the specified text file
- fix-whitespaces.sh: fixes whitespace problems (e.g. series of spaces instead of tabulations, trailing whitespaces, etc.) into the specified text file; useful to properly format files that shall committed when not using Emacs as text editor
- fix-whitespaces-in-tree.sh: fixes whitespace problems (e.g. series of spaces instead of tabulations, trailing whitespaces, etc.) in all elligible files found from specified root directory tree; useful, when operating on a fork, to properly format files once for all and commit the result so that these formatting changes remain cleanly and clearly separated from the others (just create a branch first, and tag once done)
- rename-files-in-tree.sh (just an example of pattern substitution in filenames)
- validate-yaml.sh, to validate YAML/YML files

To inspect file and directory content

- compute-checksum-in-tree.sh: computes the checksum of all files in specified tree and stores them in the specified text output file
- display-tree-stats.sh: displays simple, key stats about the specified tree (typically in order to compare merged trees)
- list-broken-symlinks.sh: lists, from any specified directory, all the symbolic links that are broken

To compare files and trees

- diff-dir.sh: performs a (single-level, non-recursive) comparison of the content of the two specified directories
- diff-tree.sh: compares all files that are present both in first and second trees, and warns if they are not identical; warns too if some files are in one directory but not in the other
- diff-json.sh: compares the two specified JSON files, once their content has been put in a canonical, sorted form (knowing notably that their keys are allowed to be in arbitrary order)

See also: Myriad's merge.sh script, a considerably more powerful tool for comparing, uniquifying, merging trees.

To copy/transfer

- transfer-tree.sh: copies (possibly through the network) a tree existing in one location to another one, in a merge-friendly manner

To remove filesystem elements

- srm (for "secure rm"): stores deleted files in a trash directory, instead of deleting them directly; see the srm full usage
- empty-trash.sh: empties the trash directory that can be filled thanks to our `srm` script
- remove-broken-symlinks.sh: removes from the specified directory all the symbolic links that are broken

Related to (UNIX) Processes

- top.sh: triggers the best "top" available, to monitor processes and system resources
- watch.sh: tracks (over time) processes that may be transient
- monitor-demanding-processes.sh: monitors endlessly the most CPU-demanding processes. Typically useful to catch an otherwise idle process that uses a full core as soon as the system gets not actively used anymore
- benchmark-command.sh: returns a mean resource consumption for the specified shell command
- list-processes-by-size.sh: lists processes by decreasing size in RAM
- list-processes-by-cpu-use.sh: lists processes by decreasing use of CPU
- kill-every.sh: kills all processes that match specified name pattern
- kill-always.sh: as long as this script is kept running, kills any process matching the specified name

Related to Network Management

- test-network.sh: diagnoses whether the various network basic facilities are functional (IP connectivity, DNS, on the LAN or on the WAN)
- monitor-network.sh: polls periodically one's local connection to the Internet, to detect (possibly intermittent) outages, notifying them if requested and helping their diagnosis
- manage-wifi.sh: starts/gets status/scans/stops the wifi support
- ip-scan.sh: scans all IPs with any specified prefix, searching for ICMP ping answers (useful to locate some devices in a local network)
- ip-examine.sh: returns locally-determined information regarding the specified IP (ping answer, reverse DNS name, traceroute and ARP)

For Development

- update-copyright-notices.sh: updates the copyright notices of code of specified type found from specified root directory (to run at each new year)
- update-all-copyright-notices.sh: updates the copyright notices of code of specified type found from specified root directory, based on the specified year range
- add-header-to-files.sh: adds specified header to specified files
- remake.sh: forces a remake of specified generated file (e.g. `.o` or `.beam`)
- list-core-dumps-in-tree.sh: locates all core dump files in current tree
- test-with-valgrind.sh: uses Valgrind to perform quality test on the specified executable
- reformat-source-file.sh: updates (in-place) the specified source file so that it complies with our conventions, be it Erlang, C/C++ or of many other file types
- reformat-source-files-in-tree.sh: reformats all supported files of various types recursively found; notably useful when forking a repository, in order to canonicalise it as a whole
- test-with-valgrind.sh: uses Valgrind to perform quality test on the specified executable
- make-code-stats.sh: evaluates very simple/basic metrics regarding source code (Erlang, Python or Java)

See also the Ceylan-HOWTOs section about build tools and the Erlang-related Ceylan-Myriad scripts.

For Version Control System (VCS)

- dci: assists efficiently and conveniently the commit of specified file(s)
- dci-all: selects recursively from current directory the files that should be committed (either added or modified), and commits them; for each of the modified files, shows the diff with previous version before requesting a commit message
- dif: shows on the console the differences between the current versions of the (possibly specified) files on the filesystem and the staged ones (i.e. the changes that might be added)
- difg: graphical version of `dif`
- dif-prev.sh: compares the current (committed) version of specified file(s) with their previous one
- difs: shows the differences between the staged files and their committed version

- st: shows the current VCS status of the specified files
- up: updates the current local version of the VCS repository
- show-branch-hierarchy.sh: shows the hierarchy of the branches in the current VCS repository
- list-tags-by-date.sh: lists, for the current VCS repository, all (annotated) tags, from the oldest one to the latest one
- list-largest-vcs-blobs.sh: lists the largest blob objects stored in the current VCS repository, sorted by decreasing size
- list-lastly-updated-vcs-branches.sh: lists, assuming to be in a clone, the VCS local or remote branches from the most recently modified one to the ones that were modified a longer time ago
- co-main.sh: performs a checkout of the main branch, regardless of its actual name (`main` or `master`)

Replacement-related

- replace-in-file.sh: replaces in specified file the specified target pattern with the replacement one
- replace-in-tree.sh: replaces, in files matching the specified pattern found from the current directory, the specified target pattern with the replacement one
- substitute-pattern-in-file.sh (possible duplicate): replaces in specified file every source pattern by specified target one
- substitute-pattern-in-tree.sh (possible duplicate): substitutes every source pattern by specified target one in all files, starting recursively from current directory
- replace-lines-starting-by.sh: replaces in specified file every line starting with the specified pattern by the specified full line

Multimedia-related

For Audio

- play-audio.sh: performs an audio-only playback of specified content files (including video ones) and directories
- listen-to-radio.sh: plays the stream of various Internet radios

-set-audio-volume.sh: sets the volume (as a percentage of the maximum one) of the target audio sink

- extract-audio-from-video.sh: strips the video information from specified MP4 file to generate a pure audio file (.ogg) out of it (original MP4 file not modified); useful, as the resulting file is smaller and less resource-demanding to playback

- resample.sh: resamples the target audio file to the specified frequency, keeping the same bitdepth
- ogg-encode.sh: encodes specified sound file in OggVorbis after having removed any leading and ending silences, adjusting volume
- convert-vorbis-to-mp3.sh: converts a Vorbis-encoded Ogg file to MP3 (sometimes it is useful to use older players)
- convert-vorbis-tree-to-mp3.sh: converts all Vorbis-encoded Ogg files to MP3 in specified tree
- trim-silence-in.sh: removes any silence at begin and end of specified file, which is updated (initial content is thus not kept)

For Voice Synthesis / Text-To-Speech

- say.sh: says specified text, based on text to speech
- record-speech.sh: records the specified speech with specified voice in the specified prefixed filename
- install-speech-syntheses.sh: installs all listed voices for speech synthesis
- get-and-install-MBROLA-and-voices.sh: installs MBROLA and corresponding voices
- test-espeak-voices.sh: tests the voices supported by espeak
- test-voices.sh: tests all supported voices

See also the Ceylan-Myriad speech support.

For Video

- convert-mov-to-x264-mp4.sh: converts specified MOV file to MP4
- set-single-screen.sh: adapts the displays so that, whether or not an external screen is connected, there is exactly one screen enabled (hence: either a native, internal one or an external one; neither both nor none)
- clone-screen-to-external.sh: clones the current, primary screen (e.g. the one of a laptop) to any external one (e.g. a LCD one), so that exactly the same content is displayed on both (same size, with no truncation)"
- fix-video-mode.sh: forces a specified video resolution

For 3D

- blender-import.sh: allows to import directly in Blender various file formats (e.g. glTF 2.0 i.e. `*.gltf/*.glb`, Collada i.e. `*.dae`, FBX i.e. `*.fbx`, IFC BIM files i.e. `*.ifc`) from the command-line, rather than doing it interactively; takes care of checking file availability and extension, launching Blender, disabling the splash screen, removing the default primitives (cube, light, camera, collection), importing specified file without having to

access menu with a cumbersome dialog requiring to go through the whole filesystem, and focusing the viewpoint directly onto the loaded objects; depends on the Ceylan-Snake Blender support

- blender-convert.sh: allows to convert, directly from the command-line, a 3D file format into a preferred one (e.g. gltF 2.0), thanks to a non-interactive Blender; depends also on the Ceylan-Snake Blender support
- install-wings3d.sh: installs Wings3D
- run-wings3d.sh: executes Wings3D
- update-fbx.sh: updates the specified FBX file so that it adopts a (hopefully) more recent version thereof (currently: FBX 7.3), typically able to be loaded by tools like Blender
- kill-unity3d.sh: kills all processes related to Unity3D (including UnityHub); if requested, also wipes out all persistent state of UnityHub and Unity3D itself (not including any user projects)

See also:

- the installation section
- Ceylan's HOWTO regarding 3D.

For Snapshots (Camera Pictures)

- rename-snapshot.sh: renames the specified picture file, based on its embedded date (used as a prefix, if appropriate), and with a proper extension; new assigned names are typically `20160703-foo-bar.jpeg`; if no filename is specified, operates on all files of the current directory
- rename-snapshots.sh: renames *all* snapshots found from current directory, so that they respect better naming conventions
- remove-snapshot-metadata.sh: removes any metadata (typically EXIF) stored in the specified snapshot(s)
- generate-lighter-image.sh: generates a lighter (smaller and of decreased quality) version of the specified image
- generate-lighter-images.sh: reduces the size of *all* image files found in current directory

For MTP Devices

- mount-mtp-device.sh: mounts any plugged, non-locked MTP device (e.g. a smartphone, an Android e-reader) to any specified mount point, otherwise to a uniquely-generated one

CCTV-related

- monitor-cctv.sh: performs online, direct monitoring from a networked security camera (CCTV), with an average quality and no audio
- fetch-cctv-monitorings.sh: fetches locally (and leaves on remote host) the set of CCTV recordings dating back from yesterday and the three days before; designed to be called typically from the crontab of your usual reviewing user
- review-cctv-monitorings.sh: allows to (possibly) fetch from server, and review conveniently / efficiently any set of CCTV recordings dating back from yesterday and the three days before
- set-cctv-internet-connectivity.sh: enables, disables or displays the current status of the Internet connectivity of any registered CCTV camera

Document-related

- check-rst-includes.sh: checks that all RST files found from the current directory are included once and only once in the sources found
- convert-rst-to-mediawiki.sh: converts the specified RST source file into a Mediawiki counterpart file
- convert-mediawiki-to-gitlab-markdown.sh: converts the specified Mediawiki source file into a GitLab Markdown counterpart file
- generate-plantuml-diagram.sh: generates a PNG file corresponding to the specified file describing a PlantUML (UML) diagram
- generate-mermaid-diagram.sh: generates a PNG file corresponding to the specified file describing a Mermaid diagram
- latex-to-image.sh: generates an image (SVG or PNG) representing the specified LaTeX formula
- affix-images.sh: pastes the first specified image onto the second one, at the specified (signed) location according to the selected direction, in order to generate the third image
- generate-pdf-from-latex.sh: generates a PDF file from a LaTeX one, and displays it
- regenerate-rst-files.sh: updates generated files from more recent Docutils files
- track-rst-updates.sh: tracks changes in the specified RST source file in order to regenerate the target file accordingly
- spell-check-rst-tree.sh: spellchecks all RST files found from the current directory
- switch-text-encoding.sh: reencodes the specified text file according to the specified encoding (by default Unicode, precisely `utf-8`)

- convert-from-ISO-8859-to-utf8.sh: converts (in-place) the specified (text) file from a ASCII format to a utf-8 one
- djvu-to-pdf.sh: converts files in the DjVU format into PDF ones

One may also rely on the Ceylan-Myriad's scripts for documentation, notably generate-docutils.sh and generate-pdf-from-rst.sh.

System-related

Admin-related

- check-filesystem.sh: checks for errors, and repairs if needed, the specified filesystem
- check-ntp.sh: reports the current, local NTP status
- set-time-and-date-by-ntp.sh: sets time and date by NTP thanks to specified or default server
- display-ups-status.sh: displays the status of specified UPS
- report-raid-disk-status.sh: reports the status of the specified RAID array (script for automation)
- report-disk-smart-monitoring.sh: reports a state change of the specified SMART-compliant disk (script for automation)
- report-ups-status.sh: reports a state change of the specified UPS (script for automation)
- record-system-settings.sh: records in-file the main system settings of the local host (typically for a durable storage in VCS to help later troubleshootings)
- get-host-information.sh (possible duplicate): returns the main system settings of the local host, and stores them in-file
- shutdown-local-host.sh: shutdowns current, local host after having performed any relevant system update
- update-locate-database.sh: updates the 'locate' database, for faster lookups in filesystems
- mount-encrypted-usb-device.sh: mounts specified LUKS-encrypted device (e.g. a USB key, or a disk), as root or (preferably) as a normal user

To install software

- install-rebar3.sh: installs the rebar3 Erlang build tool
- install-godot.sh: installs the Godot 3D engine
- install-unity3d.sh: installs the Unity3D engine

System information-related

- display-opengl-information.sh: displays information regarding the local OpenGL support
- display-raid-status.sh: displays information regarding a local RAID array
- display-battery-status.sh: displays information regarding the status of local batteries (typically for a laptop)

Convenience-related

- activate-keyboard-backlighting.sh: (des)activates (per-level) the keyboard backlighting
- disable-touchpad-if-mouse-available.sh: ensures that the touchpad (if any) is enabled iff there is no mouse connected
- toggle-touchpad-enabling.sh: toggles the touchpad activation state
- display-to-video-projector.sh: displays a screen to a video projector (various examples thereof)

Distribution-related

- update-distro.sh: updates the current distribution, and traces it
- update-distro-mirrors.sh: updates the mirrors used for packages by the current distribution
- update-aur-installer.sh: installs or updates our recommended, local AUR (*Arch User Repository*) installer, possibly because of a `pacman` update breaking the previous AUR installer instance
- install-arch-package.sh: installs a package on Arch Linux, found as either a standard Arch one (with pacman), or as an AUR package; any needed prior installation or update of an AUR installer is managed automatically

For encryption

- crypt.sh: encrypts in a standard way and as strongly as reasonably possible the specified file(s), and removes their unencrypted sources
- decrypt.sh: decrypts specified file(s) (does not remove their encrypted version)

Note also Ceylan-Myriad's support for basic, old-school ciphering, a complementary solution, and also our mini-HOWTO regarding cybersecurity.

For security

- for the management of credentials (i.e. sets of login/password pairs):
  - open-credentials.sh: unlocks (decrypts) the credential file whose path is read from the user environment, and opens it; once closed, re-locks it (with the same passphrase)
  - lock-credentials.sh: locks (encrypts) the credential file whose path is read from the user environment
  - unlock-credentials.sh: unlocks (decrypts) the credential file whose path is read from the user environment
- lock-screen.sh: locks immediately the screen
- suspend-local-host.sh: suspends immediately the local host, and ensures that it will resume in a locked state
- inspect-opened-ports.sh: lists the local TCP/UDP ports that are currently opened

One may also rely on Myriad's generate-password.sh script in order to generate safe, strong passwords (typically one per account listed based the `*-credentials.sh` scripts just above).

On a side note, we also recommend registering a domain name of one's own (e.g. `myfoobar.org`) and settting up a catch-all regarding the recipient email addresses (so that all emails sent to `*@myfoobar.org` are routed to your own personal email address).

Then, to avoid messing with your wildcard naming in terms of email addresses, you may choose first any conventional, meaningless email address prefix (e.g. `deneb`) to be dedicated to per-service communications. Next step is, in order to register to any online service (let's say it is named `http://someacme.com`), to declare to this service that your email address is `deneb-someacme@myfoobar.org` (or perhaps `deneb-someacmedotcom@myfoobar.org` if some ambiguity could remain). You would then generate a corresponding password with `generate-password.sh` and store the email/password pair among the credentials managed by the scripts above.

The advantages of this procedure as a whole are: strong and unique password, stronger pair of credentials as the login is not constant (hence less predictable[1]), and if ever you start receiving spam targeted to `deneb-someacme@myfoobar.org` then the corresponding website (e.g. `http://someacme.com`) may have some explanations to share...

See also Ceylan's HOWTO regarding cybersecurity.

Firewall configuration

- iptables-inspect.sh: lists the currently-used firewall rules for the chains of the main tables

[1] Depending on the use, the suffix for a site may be different from the site name, provided of course a translation table is kept on your side (typically in said credentials file). For example, one just has to record that for `http://someacme.com` he chose `deneb-bluetiger@myfoobar.org` (stealthier/less obvious - but then harder to incriminate a rogue website).

- iptables.rules-Gateway.sh: manages a well-configured firewall suitable for a gateway host with masquerading and various services
- iptables.rules-Minimal-gateway.sh: sets up a minimal yet functional firewall suitable for a gateway host
- iptables.rules-LANBox.sh: manages a well-configured firewall suitable for a LAN host
- iptables.rules-FullDisabling.sh: disables all firewall rules
- iptables.rules-TemporaryDisabling.sh: disables temporarily all firewall rules

See also Ceylan's HOWTO section regarding firewall management.

For smartphones

- adb-pull.sh: uploads specified local files, possibly based on expressions to the already connected and authorizing Android device
- adb-push.sh: downloads in the current directory, from the already connected and authorizing Android device, files and directories (recursively)
- set-usb-tethering.sh: sets (or stops) USB tethering on local host, typically so that a smartphone connected through USB and with such tethering enabled shares its Internet connectivity with this host

See also the For MTP Devices section.

For archive management

- archive-emails.sh: archives properly and reliably (compressed, cyphered, possibly transferred) the user emails
- Manages reference version of files, by storing them in a "vault":
  - catch.sh: stores a file in a vault directory and makes a symbolic link to it, so that even if current tree is removed, this file will not be lost
  - retrieve.sh: retrieves at least one file already stored in vault by creating link towards it, from current directory
  - update-directory-from-vault.sh: updates all files in specified directory from their vault counterparts
- make-git-archive.sh: makes a backup (as an archived GIT bundle) of specified project directory, stored in specified archive directory
- snapshot.sh: performs a snapshot (tar.xz.gpg archive) of specified directory
- list-for-backup.sh: enumerates in current directory all files, specifies their name, size and MD5 sum, and stores the result in a relevant file

Web-related facilities

- generate-html-map.sh: generates a simple HTML map with links from the available pages in specified web root
- backup-directory.sh: backups specified directory to specified backup directory on the specified server, using specified SSH port
- fetch-website.sh: downloads correctly, recursively (fully but slowly) web content accessible from the specified URL
- generate-awstats-report.sh: to trigger the generation of an Awstats report (prefer using US-Web instead)
- make-markup-shortcut-links.sh: creates shortcuts (symlinks) for the `put-*-markup.sh` micro-scripts, in order to assist a bit the user of following languages:
  - for HTML: `bold`, `box`, `cent`, `code`, `def`, `defel`, `em`, `img`, `linked`, `lnk`, `ordered`, `para`, `sni`, `strong`, `table`, `tit`, `toc`
  - for RST: `imgr`, `linkr`
  - in general: `fuda`, `newda`

For user notifications

- notify.sh: notifies the user about specified message, possibly with a title and a category
- timer-at.sh: requests to trigger a timer notification at specified (absolute) timestamp
- timer-in.sh: requests to trigger a timer notification in specified duration
- timer-every.sh: requests to trigger (indefinitely, just use CTRL-C to stop) a timer notification every specified duration
- start-jam-session.sh: starts a jam session interrupted by a notification every period, to avoid remaining still for too long
- bong.sh: plays the specified number of bong sound(s)
- beep.sh: plays a beep to notify the user of an event

Shell Helpers

To facilitate shell sessions:

- mo: shorthand for a relevant version of `more`
- hide.sh: hides specified file or directory (simply by adding a `-hidden` suffix to its filename), while unhide.sh does the reverse operation
- set-display.sh: sets the X display to specified host; if none is specified, sets it to the local one
- get-date.sh: returns the current date in our standard short format (e.g. `20210219`)

About Configuration

For keyboards

- reset-keyboard-mode.sh: resets the keyboard mode, typically should it have been modified by a misbehaving program
- set-keyboard-layout.sh: sets the X keyboard layout
- set-french-keyboard.sh: sets the keyboard layout to the French (for all X)
- set-auto-repeat.sh: enables the keyboard auto-repeat mode (to issue multiple characters in case of longer keypresses)

Of Environment and related

- unset-all-environment-variables.sh: unsets all environment variables, typically to rule out stranger account-related side-effects
- set-local-environment.sh: sets up a few installation conventions

Cooling-related

- get-temperatures.sh: to return a list of the temperatures known of the system, for various components (cores, disks, motherboard, etc.)
- apply-fan-settings.sh: to apply fan settings adequate for a silent yet well-cooled computer (see script comments to trigger it appropriately and also to grant appropriate permissions to other convenience programs)

Script-related

- protect-special-characters.sh: prevents special characters in specified expression from being interpreted by tools like sed
- encode-to-rot13.sh: returns a ROT13-encoded version of specified parameters
- default-locations.sh: detects the path of the most common UNIX-related tools (this script shall be sourced, not executed)
- term-utils.sh: defines facilities to make a better use of terminals, as a very basic, limited text user interface (this script shall be sourced, not executed)

Launch-related

- e: to edit (i.e. open potentially for updating) smartly all kinds of files; see also our bash autocomplete settings so that your shell automatically filters the arguments that it suggests whenever executing this `e` script from a command-line
- v: to view (i.e. open for reading only) smartly all kinds of files and directories

- email.sh / courriels.sh: to launch a suitable e-mail client
- launch-irc.sh: to launch a suitable IRC client

Miscellaneous

- eat-CPU-cycles.sh: generates some CPU load

Detailed description of some frequently-used scripts

wh `wh` (for "where"): searches (as a more user-friendly 'find') all files and directories matching <filePattern>, from <starting_directory> if specified, otherwise from current directory.

Usage: wh [-h|--help] [--verbose] [-q|--quiet] [--no-path] [--exclude-path <a directory>] <filePattern> [<startingDirectory>]

Options:

[-q|--quiet]: only returns file entries (no extra display); suitable for scripts (e.g. for f in $(wh -q 'foo*'); do...)

--no-path: returns the filenames without any leading path

--exclude-path DIR: excludes specified directory DIR from search

regrep `regrep`: recursive grep for the ones that lack it.

Usage: regrep [-v|--verbose] [-q|--quiet] [-f|--filenames-only] [-i|--insensitive] [-r|--restructured] [-e|--exclude ELEM]* <Expression to be found in files> [<Filter on files>]

Options:

-v or --verbose: be specifically verbose

-q or --quiet: be specifically quiet, just listing matches

-f or --filenames-only: display only filenames, not also the matched patterns, and if there are multiple matches in the same file, displays its filename only once (implies quiet); typically useful in scripts

-i or --insensitive: perform case-insensitive searches in the content of files, and also when filtering any filenames

-r or --restructured: use ReStructured text mode (skip tmp-rst directories, search only *.rst and *.rst.template files)

-e or --exclude ELEM: excludes the specified filesystem element (e.g. file or directory) from the search; as many -e/--exclude options as wanted can be specified

Example: `regrep --exclude ./foo.dat --exclude ./backup-dir -i 'little red rooster' '*.txt'`.

See also: ergrep (for searching specifically in Erlang files), pygrep (Python version), jgrep (Java version), jsgrep (Javascript version), jsxgrep (React JSX version), cgrep (C version) and cppgrep (C++ version).

srm `srm` (for "secure rm"): stores deleted files/directories in a trash instead of deleting them directly, in order to give one more chance of retrieving them if necessary. Ensures that no two file elements can collide in trash so that all contents are preserved.

Usage: `srm <files/directories to delete securely>`

See also: `empty-trash.sh`

Shell Configuration

One may also have a look at our related Ceylan-Heavy's generic, flexible shell configuration (namely `.bashrc*` files).

Windows Shell-related Support

Overview

The idea here is to go another route than using the highly-integrated graphical tools available on Windows[2], by replicating there the Unix tools that we know and like (albeit in a somehow more limited, less efficient form). Then most of the Ceylan-Hull scripts get available on Windows as well.

> **Note**
>
> We have far less experience on the Windows platform than on the GNU/Linux ones, so misconceptions may be exposed in these sections. Please tell us if it is so!

Main Choices

Here are the choices that we recommend:

- use of MSYS2 ("*Minimal System 2*"), a software distribution and a development platform for Microsoft Windows based on Mingw-w64 and Cygwin, that helps deploying code from the Unix world on Windows; this collection of tools and libraries provides a full, easy-to-use environment for building, installing and running native Windows software; we preferred the MSYS2 option over the MinGW or WSL ones
- MSYS2 reusing some elements from Arch Linux, packages are to be installed with `pacman`; we recommend to start installing the following base packages with:

[2]Many prefer instead, when it comes to programming, to rely on Visual Studio Code with various SDK (e.g. .Net ones) and many add-ons for scaffolding (like Yeoman, Yo - that relies on `npm`, a .Net package manager like NuGet , a Omnisharp server, VCS support, language support, etc, rather than creating a Unix-like environment. That's not our case!

```
$ pacman -Syu
$ pacman --noconfirm --needed -Sy man-db diffutils make gcc emacs tree mingw-w64-x86_6
```

- our shell configuration and our Emacs' one, as both support natively the Windows platform as well
- in terms of terminal, we prefer `Windows Terminal` (and `Powershell` as a last resort); the default terminal available with MSYS2 is `Mintty`, which we believe is fairly limited

More Detailed MSYS2 Information

Packages

As mentioned, MSYS2 uses `pacman` to manage its packages, and comes with 3 different package repositories:

- `msys2`: containing MSYS2-dependent software
- `mingw64`: containing 64-bit native Windows software (compiled with the mingw-w64 x86_64 toolchain)
- `mingw32`: containing 32-bit native Windows software (compiled with the mingw-w64 i686 toolchain)

Most of the mainstream package in the GNU/Linux world can be obtained, and installing them does not require administrator permissions.

We recommend the following extra packages:

```
$ pacman --noconfirm --needed -Sy mingw-w64-x86_64-aspell
 mingw-w64-x86_64-aspell-fr mingw-w64-x86_64-binutils mingw-w64-x86_64-bzip2
 mingw-w64-x86_64-ca-certificates mingw-w64-x86_64-curl mingw-w64-x86_64-diffutils
 mingw-w64-x86_64-eog mingw-w64-x86_64-evince mingw-w64-x86_64-file
 mingw-w64-x86_64-freeglut mingw-w64-x86_64-freeimage mingw-w64-x86_64-gcc
 mingw-w64-x86_64-gedit mingw-w64-x86_64-ghex mingw-w64-x86_64-gimp
 mingw-w64-x86_64-jq mingw-w64-x86_64-lz4 mingw-w64-x86_64-lzo2
 mingw-w64-x86_64-make mingw-w64-x86_64-meld3 mingw-w64-x86_64-openssl
 mingw-w64-x86_64-python3 mingw-w64-x86_64-sed mingw-w64-x86_64-sqlite3
 mingw-w64-x86_64-texlive-core mingw-w64-x86_64-texlive-extra-utils
 mingw-w64-x86_64-texlive-fonts-recommended mingw-w64-x86_64-texlive-fonts-extra
 mingw-w64-x86_64-texlive-lang-french mingw-w64-x86_64-texlive-latex-extra
 mingw-w64-x86_64-texlive-latex-recommended mingw-w64-x86_64-texlive-plain-generic
 mingw-w64-x86_64-texlive-science mingw-w64-x86_64-tzdata mingw-w64-x86_64-vlc
 mingw-w64-x86_64-wget mingw-w64-x86_64-xz mingw-w64-x86_64-zlib
 mingw-w64-x86_64-postgresql ccache
```

Paths

For a user named `my_user_name`, the user directory (home) as used by MSYS (`/home/my_user_name`) corresponds by default to `C:\msys64\home\my_user_name`.

Symbolic Links

Apparently, links are, in this Windows filesystem context, at least by default, not symbolic, but "physical" ones (*hard links*): `ln -s` *copies* the source rather than creating a symbolic link (more information: [1], [2]). At best creating a link results in a deep copy.

Solutions may exist to benefit from real symlinks, but MSYS2 would have to be specifically configured and run with extended permissions ("native priviledges").

We encountered related issues with GIT repositories that had been created in an Unix environment and that contained symlinks which, on a clone made on Windows, were replaced by a file containing the name (as a text!) of the target element at which the symlink originally pointed. Weird and unfortunate.

Other Installations

These punctual installations are not done through MSYS2.

For Erlang :

- download the Windows 64 bit binary version
- then `erl.exe` is installed as `C:\Program Files\erl-x.y\bin` (e.g. `x=24`, `y=1`), i.e., from the MSYS2's point of view, `/c/Program\ Files/erl-x.y/bin/`

More Detailed Windows Terminal Information

Windows Terminal (see its project repository) is installed by default, and supports tabs (one per running shell).

It shall be configured according to this source, by clicking on the `V` (downward arrow) then on `Settings`.

The suggested JSON sections define, in addition to the `MSYS/MSYS2` terminal entry, two MinGW-related entries that surprisingly are not compliant with our `bashrc` conventions (they enter an infinite loop apparently).

We thus recommend adding in the JSON configuration file of *Windows Terminal* only the next section:

```
[...]
   {
     "guid": "{71160544-14d8-4194-af25-d05feeac7233}",
     "name": "MSYS / MSYS2",
     "commandline": "C:/msys64/msys2_shell.cmd -defterm -here -no-start -msys",
     "startingDirectory": "C:/msys64/home/%USERNAME%",
     "icon": "C:/msys64/msys2.ico",
     "fontFace": "Lucida Console",
     "fontSize": 9
   },
[...]
```

(or, at least, to set it as the default one)

One should also define a shortcut to Windows Terminal in replacement of the MSYS2 one: enter `wt` in the Windows search bar and select "open the file location".

This should lead to a directory like `C:\Program Files\WindowsApps\Microsoft.WindowsTerminal_1.8.` in which `wt.exe` can be found, for which a shortcut may be created and placed on the desktop.

In this terminal, the right click plays the usual UNIX role of the middle-click (in order to paste).

One may:

- in the `Interaction` menu, enable "copy automatically the selection in the clipboard" so that one (and not two) right click is sufficient to copy
- in `Parameters -> Profiles -> MSYS2 -> Appearance` (not in "Color Settings" or alike, which centralises the *references* to which profiles may point), modify the colors of the MSYS2 profiles so that, notably, directory names are more readable, by example by pointing to the "One Half Dark" color setting (easier to read than the "Vintage" one)

Shell

The `.bashrc.MSYS` symlink obtained from Ceylan-Heavy may have to be fixed with:

```
$ cp .bashrc.Linux .bashrc.MSYS
```

Getting Ceylan-Hull

It will be found by our shell configuration; obtaining it is just a matter of:

```
$ mkdir -p ~/Software/ceylan/ && cd $_
$ git clone https://github.com/Olivier-Boudeville/Ceylan-Hull hull
```

Text Editor

Once Emacs has been installed and configured (see Main Choices), its graphical version can be run via `/mingw64/bin/emacs` (its console version is in `/usr/bin/emacs` and appears in the PATH generally before it).

Our `init.el` defines the `WINDOWS_HOME` environment variable to locate easily the Windows user directory from MSYS2 (knowing the aforementioned issue with symlinks).

Gotchas

`stdout is not a tty`

This error occurs when attempting to execute a program (let's name it `P`) that happens to be reported as not found.

Rather than running it as `P`, prefer specifying either `P.exe` or `winpty P.exe` (see winpty).

As many program aliases as needed can be defined in one's `.bashrc`.

Post-Install

These operations are long, but very useful:

```
$ /usr/bin/mandb --quiet
$ updatedb --localpaths='/ /c/'
```

A Word about Alternate Choices and Other Environments

MSYS

MSYS is a **collection of GNU utilities** such as `bash`, `make`, `gawk` and `grep` to allow building of applications and programs that depend on traditionally Unix tools to be present.

MSYS is a fork of an old Cygwin version (`msys.dll` is a fork of `cygwin.dll`) with a number of tweaks aimed at improved Windows integration, whereby the automatic POSIX path translation when invoking native Windows programs is arguably the most significant. MSYS is intended to supplement MinGW and the deficiencies of the cmd shell.

MSYS by itself does not contain a compiler or a C library, therefore does not give the ability to port Unix programs over to Windows nor does it provide any Unix specific functionality like case-sensitive filenames.

Users looking for such functionality should refer to Cygwin.

MinGW

MinGW is a set of **C/C++ compilers allowing the generation of Windows executables with no specific dependency** towards DLL like the ones of Cygwin; only the usual MSVC runtime is needed, which belongs to any installation of Microsoft Windows.

Compared to Cygwin:

- MinGW does not have a Unix (POSIX) emulation layer; as a result one's application needs to specifically be programmed to be able to run on Windows
- so MinGW does not require a compatibility layer DLL, and thus programs do not need to be distributed with source code; Mingw + MSYS defaults to producing binaries linked to the platform C lib

MinGW does not provide a GNU/Linux-like environment, that is MSYS and/or Cygwin.

MinGW forked from version 1.3.3 of Cygwin.

Cygwin

Cygwin has for goal to create a **complete Unix/POSIX environment on Windows**.

The Cygwin gcc and environment default to producing binaries linked to the (GPL) Cygwin DLL (`cygwin.dll` or `cygwin1.dll`).

So Cygwin allows to build Unix programs for Windows with little to no modification through a dependency on the Cygwin DLL (where MinGW does not involve such dependency but requires the program to be specifically adapted for Windows).

WSL

It seems MSYS2 is still better than WSL as it has a more native Windows experience, and offers direct Unix-like usage on Windows.

Other solutions such as Chocolatey (a package manager for Windows) exist, but we preferred to stick to a more Unix flavour.

Most of the Ceylan-Hull scripts should be operational in such a context.

See also

- the `tests` subdirectory, for a few tests of specific facilities provided here
- the `mostly-obsolete` subdirectory, for the scripts we deprecated
- our Emacs configuration (in Ceylan-Myriad)

Support

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported to the project interface (typically issues) or directly at the email address mentioned at the beginning of this document.

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the Support guidelines).

Ending Word

In the hope that Ceylan-Hull will be of use for you as well!

Hull