Ubiquitous Bash (public domain, no copyright, CC0)
Multiplatform structured programming middleware.

_ DIRECTORIES _
*) Call script (recursively with new or imported session), files, and other programs from same directory as script location.
*) Safe removal of temporary directories. Process termination trapping - SIGTERM, SIGINT, etc.

_ MODULAR _
*) Create new software projects by 'fork' script.
*) Hierarchical dependency declaration. Inclusion of either all or only desired functions and tests.
*) Override at runtime by '_local/ops.sh' and similar files.

_ OPERATING SYSTEM and VIRTUALIZATION _
*) Bootable OS, LiveISO, LiveUSB building and customization simultaneously through ChRoot, rsync, BIOS, UEFI, VirtualBox, Qemu, convertible to/from Docker.
*) Hibernation and/or persistent filesystems within LiveISO, LiveUSB, as native bootable alternative to virtualization provided 'save state' or 'nonpersistence'.

*) Application virtualization (and graphical applications) through several fallback backends ('_userChRoot', '_userVBox', '_userQemu', '_userDocker') .
*) Guest program to launch, translated fileparameters, and network filesystem mappings indicated by batch and shell scripts from 'hostToGuestISO' temporary CDROM disc file.

*) Groups of virtual machines portability forced by '_labVBox' (notably useful with PFSense to simulate complete WAN network configuration at high fidelity).

*) Consistent apparent location of project directory through '_abstractfs' (notably used by 'arduinoUbiquitous' to make firmware builds portable).
*) Home directory portability, and persistence or nonpersistence, forced by '_fakeHome' (notably used by 'webClient' to force browser instances).

*) Legacy OS portability through '_winehere' and '_dosbox' .
*) MSW compatibility integrated by portable 'ubcp' 'ubiquitous bash cygwin portable' and 'anchor' 'shortcut' batch files simultaneously interpretable as bash script (due to interleaved 'comment' characters).

*) Linux Kernel configuration review, tradeoffs, and recommendations ('_kernelConfig_desktop', '_kernelConfig_mobile', '_kernelConfig_panel').
*) Hardware support (eg. 'Lenovo x220t', 'Huion h1060p', etc) .

_ INTER-PROCESS COMMUNICATION _
*) Multiple-input multiple-output directory pipes, triple buffers, and searchable 3D coordinate spaces, interfacing between programs through 'MetaEngine' .

*) 'Broadcast' software bus 'queue' more similar to hardware bus (eg. 'CAN bus', 'UART', 'I2C', etc) emulating 'shared pair of wires'.
 *) Pipes automatically recreated through '_demand_broadcastPipe_aggregatorStatic', '_aggregator_read', '_aggregator_write' .
 *) Triple buffers directory through '_demand_broadcastPipe_page', '_page_read', '_page_write' .
 *) Database IPC through '_db_read', '_db_write' (may track Virtual Machine or other simulated hardware UART serial ports to connect through software IPC).
 *) Network adapters through '_aggregatorStatic_socket_unix_server', '_aggregatorStatic_socket_unix_client', '_aggregatorStatic_socket_tcp_server', '_aggregatorStatic_socket_tcp_client', '_page_socket_tcp_server', '_page_socket_tcp_client', '_page_socket_unix_server', '_page_socket_unix_client' .

_ DEV _
*) Situational awareness command-line Bash and Python prompts after '_setupUbiquitous' .
*) Equation solver ('c', '_clc') added to Bash and Python prompts after '_setupUbiquitous' ('Qalculate' and 'GNU Octave' backends).
*) Python to Bash and Bash to Python bindings with Ubiquitous Bash functions (eg. ' print(_getScriptAbsoluteFolder()) ')
*) Build dependencies, compiling, and calling other build systems (eg. cmake).

_ REMOTE _
*) NAT public services and inbound SSH fallbacks. AutoSSH robust configuration, automatic restart, tested over multiple years.
*) SSH automatic multi-path and multi-hop.
*) VNC to existing (ie. '_detect_x11') and new (ie. 'x11vnc') display sessions, through SSH commands without installed service.

*) Cloud - 'rclone', 'aws', 'gcloud', 'digitalocean', 'linode', etc (notably cloud services may efficiently build, test, and distribute software).

Much more functionality exists than listed here. At approximately ~1500 function declarations, ~20000 lines of shell script code, ~1MB (~1 million characters), years of intense development, and years of thorough testing, Ubiquitous Bash exists to resolve many very significant and substantial software issues.

Demonstrations

Situational Awareness Command-Line Bash and Python Prompts

(pictured right)

Linux Application Virtualization

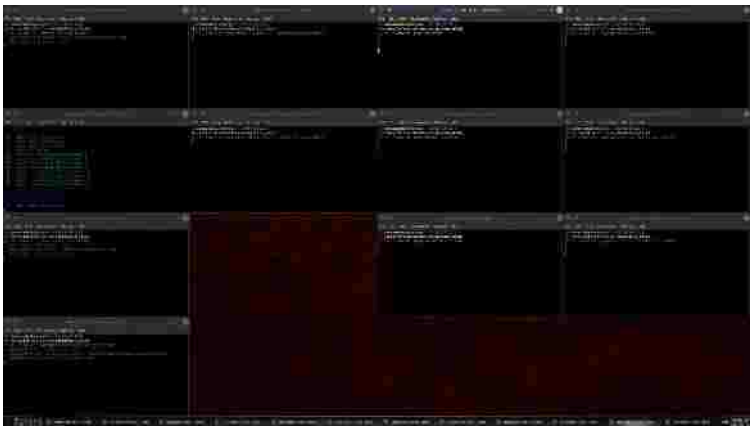
```
./ubiquitous_bash.sh _userQemu leafpad ./CC0_license.txt
```

MSW Application Virtualization

```
./ubiquitous_bash.sh _userVBox notepad.exe ./CC0_license.txt
```

Queue Inter-Process Communication

(pictured right)



_test-shell

```
 './ubiquitous_bash.sh' '_test-shell' '2>/dev/null'
#  Sanity...
#  PASS
#  Permissions...
#  PASS
#  Argument length...
#  PASS
#  Absolute pathfinding...
#  PASS
```

Examples

DIRECTORIES, MODULAR

BOM_designer

Hierarchical text-based Bill-of-Materials. Finds all all files with extensions '.lbom.csv', '.lbom.txt', '*.lbom' . Compiles a consolidated list.

gEDA_designer

Designer. Extensively automates integration and manufacturing specification files (eg. PDF, CAD model, 'gerber', PCB photolithography masks, etc).

scriptedIllustrator

From shell script to interleaved self-modifying shell script and markup of 'html', 'pdf', 'mediawiki', 'markdown', etc.
https://github.com/mirage335/scriptedIllustrator

OPERATING SYSTEM and VIRTUALIZATION

arduinoUbiquitous

Both '_abstractfs' and '_fakeHome' contain 'Arduino', related programs, and libraries, to create firmware projects with portability comparable to "makefile" or "cmake" projects. External debug tool interfaces (eg. 'gdb', 'ddd') are also integrated.
https://github.com/mirage335/arduinoUbiquitous

webClient

Forces browser profile portability and multi-instance through '_fakeHome'.
https://github.com/mirage335/webClient

freecad-assembly2

FreeCAD assembly2 and a2plus module portability through '_fakeHome'.
https://github.com/mirage335/freecad-assembly2

kit-raspi

RasPi and x64 bootable OS, LiveISO, LiveUSB building and customization mostly through '_chroot', 'cp -a', 'rsync' .
https://github.com/mirage335/ubiquitous_bash/tree/master/_lib/kit/raspi

INTER-PROCESS COMMUNICATION

metaBus

Reference implementation illustrating connecting multiple programs through 'MetaEngine' .
https://github.com/mirage335/metaBus

DEV

pcb-ioAutorouter

Compiles and contains installation of 'pcb' with patch for 'autorouter' compatibility.

REMOTE

CoreAutoSSH

Remote logical network configuration . Automatic SSH multi-path and multi-hop.
https://github.com/mirage335/CoreAutoSSH

Usage

```
./ubiquitous_bash.sh
./ubiquitous_bash.sh _test
./ubiquitous_bash.sh _setup
```

Projects using Ubiquitous Bash as a git submodule can be created by 'fork' script. Examples of common modifications to such projects are available at '_lib/kit' directory. Project name, and developer name, must be set by editing 'fork' script appropriately.

```
# Move 'fork' to a different directory and edit appropriately.
./fork
```

Python

```
./ubiquitous_bash.sh _python

print(_getScriptAbsoluteFolder())
_bin("_getAbsoluteFolder .")
_bin("_getAbsoluteLocation .")

_clc('1 + 2')
_qalculate('1 + 2')
_octave('1 + 2')
print(_octave_solve('(y == x * 2, x)' ))

_bash( '-i' )
_python
exit()
exit
exit()
```

Support

Ubiquitous Bash is supported on an *urgent basis*, nominally *immediate*, due to long established dependability and uses. Any support request will be attended to ASAP. Please do not hesitate to contact maintainer "mirage335" by any means.

Bug reports, feature requests, forks, and especially pull requests, are highly welcome. Please keep in mind "defense in depth" and explicit tests are preferred measures to ensure against regressions. Ubiquitous Bash GitHub repository is monitored frequently if not in real time.

Design

Entry points for developers are described here. To understand Ubiquitous Bash, often it will be necessary to search for these bits of shell code in 'ubiquitous_bash.sh' or other related files.

```
_findFunction() {
        find . -not -path "./_local/*" -name '*.sh' -type f -size -3000k -exec grep -n "$@" '{}' /dev/null \;
}
```

DIRECTORIES

Script absolute 'location', script absolute 'folder', unique "sessionid", will be set, and temporary directories created, as specified by structure/globalvars.sh .

```
export sessionid=$(_uid)
export scriptAbsoluteLocation=$(_getScriptAbsoluteLocation)
export scriptAbsoluteFolder=$(_getScriptAbsoluteFolder)
[[ "$tmpSelf" == "" ]] && export tmpSelf="$scriptAbsoluteFolder"
export safeTmp="$tmpSelf""$tmpPrefix"/w_"$sessionid"
```

New "sessionid" is commonly obtained to separate temporary directories. Temporary directories are created by '_start', removed by '_stop'. Process termination signal (eg. SIGTERM, SIGINT) calls '_stop'.

```
_userFakeHome_procedure() {
        export actualFakeHome="$instancedFakeHome"
        export fakeHomeEditLib="false"
        _fakeHome "$@"
}

_userFakeHome_sequence() {
        _start

        _userFakeHome_procedure "$@"

        _stop $?
}

_userFakeHome() {
        "$scriptAbsoluteLocation" _userFakeHome_sequence "$@"
}

"$scriptAbsoluteLocation" _userFakeHome /bin/true || _stop 1
```

MODULAR

After 'fork' script has created a new project, new code may be added to the usual files in the '_prog' directory among other places. These and other desired shell script fragments are 'compiled' .
Typical program entry point is usually '_main()' defined through '_prog/program.sh' file.

```
./compile.sh
```

Another possibility is to edit 'lean.sh' directly. Such is strictly monolithic and inconvenient to upgrade, not recommended for most software projects which tend to become quite large rather quickly.

```
export ub_setScriptChecksum_disable='true'
#...
#####Entry
#...
_main "$@"
```

Any function name with '_' as first character, can be called externally as first parameter to script.

```
./ubiquitous_bash.sh _echo echo
```

'Complete' 'Ubiquitous Bash' '_test' will test (and install for some 'Linux' 'distributions' ie. 'Debian') such complicated dependencies as 'VirtualBox' and 'Docker'.

```
./ubiquitous_bash.sh _test
./ubiquitous_bash.sh _setup
```

Version

v3.113

Semantic versioning is applied. Major version numbers (v2.x) indicate a compatible API. Minor numbers indicate the current feature set has been tested for usability. Any git commits not tagged with a version number may be technically considered unstable development code. New functions present in git commits may be experimental.

In most user environments, the latest git repository code will provide the strongest reliability guarantees. Extra safety checks are occasionally added as possible edge cases are discovered.

Conventions

*) Assign ports in ranges 55000-65499 and 50025-53999 to specialized internal servers with opsauto procedures.
*) Strictly single use ports are by default assigned in range 54000-54999 .

Safety

*) DANGER: Do NOT 'open' 'loopback' backends (eg. _openImage, _openChRoot, _openVBoxRaw) as read/write (ie. 'edit' instead of 'user') if reboot has occurred while open. Wrong 'loopback' 'device' may be overwritten. All 'loopback' backends are intended ONLY for developers or for building 'operating system images'. End-user activity should never cause a call to a loopback device, even indirectly.
 *) Nevertheless, significant safety checks are in place to reduce risk of data loss in any way other than 'loopback' 'device' conflict. If the specific data referenced by a 'loopback' 'device' is time-consuming to recreate, consider '_bupStore' as 'version control' 'backup'.

*) VirtualBox raw image backend is complicated, possibly fallible, and may not be compatible with MSW hosts. End-users should instead use a file converted to 'VDI', with a copy of the 'raw' file for Qemu as a fallback if necessary.

*) Cloud backends may still be experimental if workable at all.

*) Docker backends have not proven fundamentally or frequently useful, and thus may not be recently tested for interoperability (eg. with VirtualBox, UEFI, LiveISO, LiveUSB etc).

*) Obviously, safeRMR is not foolproof. Use this function to guard against systematic errors, not carelessness.
*) Test any modifications to safeRMR in a safe place.
*) A few commands and internal functions, eg. "type" and "_timeout", are used to ensure consistent behavior across platforms.
*) Interdependencies between functions within the ubiquitous_bash library may not be documented. Test lean configurations in a safe place.

*) ChRoot based virtualization by itself does not provide any security guarantees, especially under Linux hosts. Destruction of host filesystem is possible, especially for any guest filesystem that has been bind mounted.
*) RasPi image must be closed properly (if opened by chroot) before flashing. Doing so will re-enable platform specific "/etc/ld.so.preload" configuration.
*) Images opened in docker must be closed properly to be bootable.

*) Launching "user" ChRoot as root user has not yet been extensively tested.

*) Shared resource locking (eg. disk images for ChRoot, QEMU, VBox) is handled globally. Do NOT attempt to launch a virtual machine with one backend while still open in another. Likewise, separate virtual machines should be handled as separate projects.

*) Do NOT add empty functions anywhere, as this will cause the script to crash before doing anything. At least include a do-nothing command (ie. /bin/true).

*) Each project using ubiquitous_bash should incorporate it statically. Dynamic system-wide linking with other projects is STRONGLY discouraged. However, projects based upon ubiquitous_bash might be suitable for system-wide installation if designed with this in mind.

*) Anchors pointing at non-existent functions may cause themselves to be executed by the "ubiquitous_bash.sh" script they point to - resulting in an endless loop. However, the worst case scenario is generally limited to user log file spam.

*) WARNING: Do NOT export specimen directories as part of Eclipse projects. ONLY export project files, and reference specimen directories relative to the workspace path variable. Symlinks to temporary and scope directories may not be intended for the recursive copy that would result. New users do not need to worry about this until they know what it means.

*) Atom packages are not necessarily safe for portable operation. At least "Command Toolbar" makes reference to absolute file paths. Atom packages installed with "apm", as would be done for git submodules, also make reference to absolute file paths. Atom cannot be relied upon as a general purpose project specific IDE.
	For "Command Toolbar, a workaround is to delete, and manually edit, "_lib/app/atom/home/.atom/command-toolbar.json" .

Future Work

*) Voice commands using Pocket Sphinx and limited vocabulary. Specifically NOT a 'digital assistant'.
*) Voice feedback.

*) Replace all use of 'losetup' with 'dmsetup' uniquely named 'dm-linear' 'devices', after ' blockdev --getsz ', as with 'packetDrive' .

*) Demonstrate ability to preserve less dependable SSDs (ie. typical SD Cards) by automatically detecting usable '/dev/shm' as "$metaDir" (derived from "$metaTmp") location.

*) Self-contained SAMBA server would provide useful virtualization compatibility guarantees if tightly integrated. QEMU seems to already include a solution using similar methods.

*) Self-contained SSH server (not requiring full virtualization images) would allow full self-testing of SSH procedures.

*) Merge HostedXen, and other related virtualization methods into ubiquitous bash.
*) Support shutdown hooks through init daemons other than systemd.
*) Service/cron installation hooks.

*) Support Xen (xl) as a virtualization backend.
*) Support LXC as a virtualization backend.

*) Integrate AppImage build scripts.

*) Investigate Kubernetes integration - https://kubernetes.io .
*) Investigate HyperKit relevance - https://github.com/moby/hyperkit .
*) Investigate RancherVM relevance - https://github.com/rancher/vm .
*) Investigate LXD relevance - https://www.ubuntu.com/containers/lxd .

*) Document FireJail and AppImage examples.

*) Set up host architecture specific hello binary compilation and switching.

*) Add type check to open/close functions (if not already present), preventing, among other things, collisions between virtualization platforms.

*) Self-hosted snippet manager. Possibly as 'Konsole' shortcuts.

*) Graphical DRAKON and/or Blockly/SigBlockly examples.

*) Nested userChRoot in userChRoot . Beware, this bizarre scenario might cause guest corruption, a mess of mounts, or worse.

*) Hard and soft cpu, memory, swap, I/O, and storage limits on all subprocesses independent of full virtualization.

*) Automatically adjust limits and priorities based on system latency impacts correlated to program operation.

*) Demonstrate backgrounding in _main as a means to launch multiple daemonized services under control of a single script instance.

*) MSW(/Cygwin) may benefit from reimplementations as simple 'C' programs, due to apparently relatively high CPU usage caused by "bash" script loops under Cygwin. However, due to the absolute portability of 'reference' "bash" implementation, expense of any rewrites, and very limited functionality needed by MSW, such should only be done in an unlikely extreme abundance of resources and/or extreme necessity.

*) MSW(/Cygwin) 'tripleBuffer' "bash" implementation may benefit from a simple 'C' program to create/read/write(/delete) files as shared memory files. A precompiled binary may be usable.

*) Dynamically reading/writing from/to multiple pipes/TCP/sockets (new pipes added/removed without resetting existing pipes) may be possible for a binary C program with multiple threads. Such a binary program must nevertheless be built by 'Ubiquitous Bash', provided with 'Ubiquitous Bash' (_bin/) , and architecture overridden by 'Ubiquitous Bash' (ie. precompiled binaries for '-amd64' , '-armel' , etc , must be aliased by a single shell script function).

*) Reference packetization implementation.

*) Reference peripheral identification implementation.

*) Simple one-input-multiple-output shared memory 'tripleBuffer' channel to 'publish' high performance data streams (eg. VR compositor frames) by most recent page.

Known Issues

*) Typical TCP/IP and related software cannot be configured for resilience under packet corruption, heavy packet loss, multi-second latency, or address/ap roaming. SSH in particular may fail erratically due to packet corruption. A proxy handling these issues through named pipes and tcpwrappers may not be possible, nor may it be feasible to workaround such SSH failures. Best to upload "$scriptAbsoluteLocation" and call entire function at server side with only one remote SSH command.
https://stackoverflow.com/questions/28643392/bash-scripting-permanent-pipe

*) Some ChRoot mounting functions are in fact generic, and should be renamed as such after audit.
*) Nested virtualization needs further testing and documentation, especially beyond QEMU.
*) ChRoot close function might have a path to exit true while mounts are still active.

*) KWrite under ChRoot may lock up the mounts, preventing _closeChRoot from working. Error messages suggest PulseAudio is not working normally. Nevertheless, cleanup seems to take place through the systemd hook upon shutdown.
*) LeafPad sometimes fails to launch from Docker container, apparently due to X11 issues. Not a problem for all graphical applications apparently (eg. 'xmessage').

*) BashDB is given a "frame 0" command to show large file source code window in emacs "realgud".

*) Order of termination does not strictly seem to preserve parent scripts with uncooperative children, when "_daemonAction" is used to terminate all daemon processes. This has the effect of allowing uncooperative children to interfere with logging and stop processes the parent may need to implement in non-'emergency' situations.

*) SSH over Tor through "_proxyTor_direct" has not been tested as is under MSW/Cygwin hosts. With the adoption of socat instead of netcat, this should be straightforward.

*) An error has been thrown to standard error upon shutdown in some rare cases. Preventative measures are now believed to be effective, however, as the error is intermittent, bug reports are encouraged. Without sufficiently concise and relevant information, this may not be resolvable.

```
        realpath: .../ubiquitous_bash/w_.../.ssh/.../cautossh: No such file or directory
        find: '.../ubiquitous_bash/w_.../.ssh/.../w_...': No such file or directory
```

Reference

https://developer.apple.com/library/content/documentation/Darwin/Conceptual/index.html
https://en.wikipedia.org/wiki/Software_architecture
https://en.wikipedia.org/wiki/Middleware

https://bugs.eclipse.org/bugs/show_bug.cgi?id=122945
https://superuser.com/questions/163957/is-eclipse-installation-portable

https://www.suse.com/support/kb/doc/?id=7007602
https://en.wikipedia.org/wiki/CPU_shielding
https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cpu-partitioning/irqbalanced
https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cpu-partitioning/cpusets

https://en.wikipedia.org/wiki/Scene_graph

https://github.com/PipelineAI/pipeline

https://wiki.bash-hackers.org/scripting/obsolete

https://novnc.com/info.html
https://novnc.com/screenshots.html
https://docs.unrealengine.com/en-US/InteractiveExperiences/UMG/UserGuide/WidgetTypeReference/WebBrowser/index.html
https://www.highfidelity.com/blog/vnc-in-vr-synchronized-virtual-desktops-49bc4fc428e7

https://github.com/shellinabox/shellinabox

https://uploadvr.com/virtc-virtual-desktop/

https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-web-client-admin
https://www.youtube.com/watch?v=NYEyyVDsapw

https://en.wikipedia.org/wiki/HashiCorp

https://en.wikipedia.org/wiki/Terraform_(software)
https://www.terraform.io/docs/language/index.html

https://en.wikipedia.org/wiki/Vagrant_(software)

https://learn.hashicorp.com/tutorials/packer/docker-get-started-provision

https://wiki.debian.org/Vagrant
        'Libvirt is a good provider for Vagrant because it's faster than VirtualBox and it's in the main repository.'

https://unix.stackexchange.com/questions/297792/how-complex-can-a-program-be-written-in-pure-bash

https://docs.docker.com/engine/admin/volumes/volumes/
        'While bind mounts are dependent on the directory structure of the host machine, volumes are completely managed by
Docker.'

https://winaero.com/pin-a-batch-file-to-the-start-menu-or-taskbar-in-windows-10/
        ' cmd /c "full path to your batch file" '

Credit

*) Thanks to "rocky" for workaround to bashdb/emacs issue - https://github.com/realgud/realgud/issues/205#issuecomment-354851601
.

Included Works

*) MAKEDEV . Obtained from Debian Stretch. For all details, see _presentation/MAKEDEV .
*) GoSu . See https://github.com/tianon/gosu . Binaries and signatures may be included in repository. License, GPLv3 . License text at https://www.gnu.org/licenses/gpl-3.0.en.html .
*) Firefox .

*) geth
*) parity
*) ethminer

Bundle

Larger files (eg. emacs configuration, blockchain) are not included with 'ubiquitous bash' directly, or as a submodule. If needed, 'clone' to '_bundle' directory (which is ignored).
git clone --depth 1 --recursive git@github.com:mirage335/ubiquitous_bash_bundle.git _bundle

Copyright