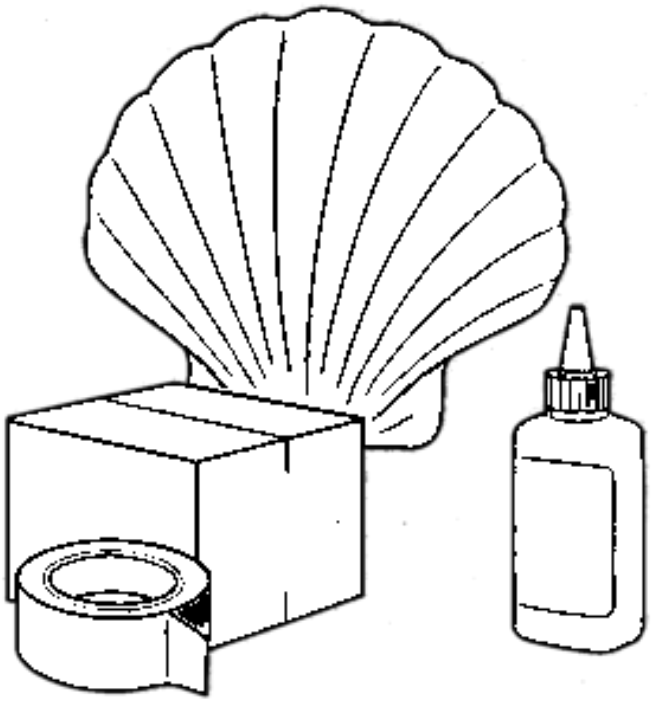
Using “ubiquitous_bash”

ALCHEMY Glue Book



Contents

Preface **1**

What is bash ? . . . 1

What is "ubiquitous_bash" ? . . . 3

What you Gain from "ubiquitous_bash" 4

What next? . . . 5

Formal Table Of Contents **6**

Tutorials **7**

Fork Derivative Project . . . 8

. . . 8

Framework - Virtualization **9**

Chapter3 **Obscure - Virtualization **blah blah blah blah blah**** **10**

Preface

Using “ubiquitous_bash”

ALCHEMY Glue Book

Figure 1: Shell, Glue, Box, Duct Tape

What is bash ?

What bash shell does.

The bash shell is essentially a chat program. It reads commands from you typing in a chat window called a ‘terminal’ or ‘command prompt’, or it reads commands from a script. Then the bash shell will call the OperatingSystem kernel (eg. Linux, or the MSWindows kernel) to run the program requested by that command.

All programs on a modern computer OperatingSystem are run this way: typing `'C:\Windows\explorer.exe'` at such a command-line shell on MSWindows can (if not already running) start the graphical shell, from which you get your ‘Start Menu’ . Equally, if you have typed `'C:\Program Files\Firefox\Firefox.exe'` as the command in the properties of a shortcut file, with the graphical shell you can double click that shortcut file, which will cause the graphical shell to ask the OperatingSystem kernel to run that program. As you may have guessed, just as command-line shells can run graphical shells, graphical shells can run command-line shells, which is usually how you get your interactive ‘chat’ window with a command-line shell such as bash.

You can also boot a computer directly to a command-line shell without a graphical shell, more than decades ago this was how most people used computers. In modern times, booting directly to only a command-line shell is something you don’t need to worry about: in situations where this is common, such as with servers, or over a serial port to an experimental hardware device, you will be given simple, clear instructions to follow. Usually you will run a given command to run in your own bash shell, which will then connect your interactive ‘chat’ to ‘talk’ to that remote computer instead until you type the ‘exit’ command.

If you understood all that, then you can now imagine how the bash shell, a graphical shell, or any other graphical shell, is always running the same special binary instruction, which interrupts and asks the OperatingSystem kernel, to run a program at the file location and with parameters, defined entirely by the same short string of human readable text. Were you able to 'talk' the OperatingSystem kernel directly, you would be able to type out `'C:\Program Files\Firefox\Firefox.exe'` to run your web browser, equally the same as typing that at a shell, or double-clicking the relevant shortcut file in a graphical shell with the same text string. Note this does imply both command-line and graphical shells are something of a file manager: that is absolutely the case, most file managers are indeed shells capable of running programs equally the same way.

If the concept that a shell, both a command-line shell as well as a graphical shell, is merely a convenient user-friendly program, most often a file manager app, to ask the OperatingSystem kernel to run another program, still leaves you with some questions, then it is a good idea to ask those questions of an AI LLM .

One of the best free AI LLM chat services is OpenRouter .

https://openrouter.ai/chat

Maybe the best AI LLM model to chat with about technical questions is the open model `Llama 3.1 Nemotron Ultra 253b` .

https://openrouter.ai/chat?models=nvidia/llama-3.1-nemotron-ultra-253b-v1:free

How to see a bash shell interactive 'chat'.

If you're using a Linux/UNIX computer distribution/OperatingSystem, open a program like 'xterm' or 'konsole', or just search for some program from a menu 'terminal'. If what you see now is a short string like `user@hostname: ~#` followed by a text cursor, where you can type a command such as `echo happy`, then in the vast majority of cases: you are looking at a `bash` interactive 'chat'. You can type all of the same commands here that you could write to a file named 'file.sh' and run as a program, and you have the full power of a Turing-complete programming language, in a language particularly designed to easily call other programs. You can even run graphical programs, such as your favorite web browser.

If you're using MSWindows, go to the start menu, search for something like 'cmd' or 'powershell', run that. Not quite bash, but another shell rather similar to bash, made for the same purpose. You can type a command such as `echo happy` here too, the difference is a bash shell has a lot of more useful commands and a much more convenient syntax for such things running one program after another. You can install a bash shell on MSWindows by quickly installing "extendedInterface" (an "ubiquitous_bash" based project) as explained under the next heading, and then you will be able to click a '_bash' shortcut to get a real bash shell, same as a Linux/UNIX computer.

```
user@hostname: ~# echo happy
happy
user@hostname: ~# _
```

What to do with a bash shell.

If this seems intimidating, don't worry, there are only a few lines of code you will use often.

Code	Description	Example
/home/username	The directory /home/username . UNIX/Linux directories all start at `/` , much like `C:\` .	

Code	Description	Example
./	The current directory . Often used to call a script.	./ubiquitous_bash.sh
../	The parent directory .	cd ../
cd	Change to a different directory.	cd /home/username ; cd
ls	List files in directory.	ls ./
less	Read a file in a directory. Page up/down to scroll, q to quit.	less ./ubiquitous_bash.sh
git	Version control software. Saves, loads, downloads, uploads, newer and older versions of files in the current direct directory.	git status

You can always ask an AI LLM model how to do something you want to do in a bash terminal. The vast majority if not all general purpose AI LLMs, even the smallest locally runnable AI LLMs such as Llama 3.1 8b, are perfectly reliable explaining and performing basic bash shell commands. This is because bash commands are extremely common in online discussions about computers, distributions/OperatingSystems, etc, and so will have been included in the basic predict-the-next-token pretraining used to give a basic understanding of human language to AI LLMs.

If you want to learn more about what distributions/OperatingSystems are, how a computer goes from power on, to BIOS/UEFI/firmware, to bootloader, to kernel, to init, to CLI shell or GUI shell, to graphical programs, to how bash shell commands call on the kernel to run programs the same way all programs are run under a distribution/OperatingSystem, then chatting with an AI LLM about each of these subjects is a fun and interactive way to learn.

What is "ubiquitous_bash" ?

Mostly "ubiquitous_bash" is a set of helper functions for the bash shell. When you define a helper function in bash by typing something like this in an interactive 'chat' or script file:

```
_helperFunction() {
    echo "Hello world!" "$@"
}
```

You can call that function as a subsequent command in an interactive 'chat' or script file:

```
_helperFunction "Yay!"
```

Which will of course simply output this:

```
Hello world! Yay!
```

Mostly the "ubiquitous_bash" project just provides a lot of these functions, to reliably find and call other programs with the correct parameters.

You can install the self-contained "ubiquitous_bash" project (~50MB) which includes the ~2MB 'ubiquitous_bash.sh' script itself, on a UNIX/Linux computer just by downloading it. To download, just type another one of those interactive 'chat' commands:

```
git clone --recursive https://github.com/mirage335-colossus/ubiquitous_bash.git
```

You can make the functions available to type in your interactive shell, and also get the fancy colorful visualPrompt, by running the '_setupUbiquitous' function:

```
cd ./ubiquitous_bash.sh
./ubiquitous_bash.sh _setupUbiquitous
```

MSWindows users can install "ubiquitous_bash" simply by downloading and running the latest "extendedInterface" installer. Just make sure you have at least ~2GB free space on `"C:\"` . When done, you will find a `C:\_bash.bat` file which can open a bash interactive 'chat' window, and a shortcut file to it which you can pin to the Start menu.

https://github.com/mirage335-colossus/extendedInterface/releases/download/build-15624422433-1/extIface.exe

What you Gain from "ubiquitous_bash" .

By getting started with "ubiquitous_bash", you can start with well-maintained commands proven to meet meet the needs of use cases and subsequent commands, letting you focus on the logic of your scripts and on exploring new platforms or projects.

You can hit the ground running even faster with "ubiquitous_bash" than without, and that little early investment will pay off in reliable, portable workflows—far more efficient than reinventing each part yourself.

The "ubiquitous_bash" framework gives a ready-made environment for stabilizing and experimenting with software from an easy-to-use interactive shell scripting chat window, or through the popular VSCode IDE, etc. Rather than iteratively figuring out and maintaining every parameter, workaround, and multi-step configuration yourself, you can immediately tap into conveniences that the project already provides:

- Cross-platform scripting – You can write once and run on Linux, or even MSWindows . Double click the exact same 'batch' file on UNIX/Linux and MSWindows to open a '_bash' terminal or run a function in your own shell script, calling whatever programs with any files and parameters you want.
- AI-assisted command-line, development, processing - CLI Codex available every command-line, sophisticated Codex prompts, with optional control of the sandbox harness that really works, nearly as capable as WebUI Codex on your local system to make things work no matter what.
- Local AI help and automation - Install and run the powerful 'Devstral-Small' model locally with Vision on 16GB VRAM (~20tokens/sec) or CPU . Developers can also have 'Llama-augment' (~60tokens/sec) agentically, locally, process pipelines. Never worry you won't have your AI programming helper with you, and get that download URL no matter how the format changes.
- Lightweight containment – Python "virtualization" and directory virtualization '_fakeHome' , 'abstractfs' , keep your environment tidy, separating applications, stabilizing file locations, without needing full-blown virtual machines.
- Full distribution/OperatingSystem builds or app-level virtualization – When you're ready to just build your own custom distribution/OperatingSystem, _openChRoot to get started without fumbling loopback devices, and launch sandboxed applications with fileparameter translation under your favorite backend - _userVBox, _userQemu, _userChRoot, _userDocker , and more. Even DosBox and Wine installations can be kept portable and convenient.
- Limitless Inter-Process-Communication and pipelines – "Queue" and "MetaEngine" offer ad-hoc communication and multi-input-multi-output pipelines so you can string together commands in advanced ways. Be ready for a future of connecting applications to more applications.

What next?

Before proceeding to the tutorials, etc, you should have at least a basic proficiency with the bash shell, sufficient to navigate to subdirectories with `cd` , list files with `ls` , read a file with `less` , navigate to parent directories with `cd` , and run programs and scripts by calling `./script.sh` , and ideally it would be best to understand the use of `chmod u+x` as well .

If you haven't been able to get a bash shell working on your own computer, you can ask an AI LLM to 'Please pretend roleplay this is a bash shell terminal.' . An AI LLM model can be used to write a prompt to get an AI LLM model to do such roleplay. Beware the resulting roleplay may have some gaps and inaccuracies, but will almost always be highly consistent, more than sufficient to practice quickly exploring what is available in a command line environment.

Please write a prompt to get an AI LLM to roleplay a bash shell terminal.

Prompt:
"You are now going to act as a Bash shell running on a typical Linux machine.
• Begin every response with the prompt string '$' (dollar-sign and a space).
• Treat whatever I type after the prompt as a command entered at the terminal.
• Return exactly what the real Bash interpreter would output for that command, followed by a new '$' prompt on the next line, ready for more input.
• If a command would normally produce no output, just show the next prompt.
• Do not add commentary, explanations, or code fences—only the raw terminal output.
• Assume standard Ubuntu-like defaults and that all common GNU utilities are installed.
• If the command tries to access the internet, requires elevated privileges, or would be destructive, respond with the same error or permission message Bash would show.
Begin when you read my first command."

Although not all AI LLM models may be suitable for such roleplay, at least some open AI LLM models seem sufficiently consistent for shell practice.

https://openrouter.ai/chat?models=meta-llama/llama-3.1-8b-instruct:free

Formal Table Of Contents

1. # Tutorials
2. Framework – Virtualization

Tutorials

Fork Derivative Project

asciinema ... animated png ? ...

Framework - Virtualization

Functions... occasional call graph...

_userVbox _abstractfs _fakeHome _useQemu

Chapter3 Obscure - Virtualization **blah blah blah blah blah**

vBoxConvert, hardware config, etc ?

blah blah blah blah blah

blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah

blah blah blah