develeap
education

Linux and Bash Proficiency Exam

Prerequisites

Before joining the bootcamp, you must be comfortable with the Unix command line and able to perform basic tasks like file manipulation and writing scripts. This exam ensures you have the necessary skills.

To prepare:

- Many excellent resources exist online for learning Bash - we recommend the free ebook "The Linux Command Line"
- Install Ubuntu and use it as your primary OS (dual boot recommended)
- Test your script in a Linux environment - this will be your working environment throughout the course

Important: While AI tools can assist learning, using them without understanding the solution is counterproductive. Success in this bootcamp requires genuine understanding of Linux and Bash fundamentals. If you pass this exam, the next stage of the process includes discussing your script and explaining your implementation choices - you must understand what you submit.

You should fulfill **all requirements** in these instructions and submit your script via the link in your exam email. If you're unsure about any aspect of the task, use your best judgment and make reasonable decisions based on standard Linux/Bash practices.

Task Instructions: File Unpacking Script

Create a Bash script named `unpack.sh` that handles multiple compressed files and can traverse directories to unpack archives.

Script Synopsis

```
unpack [-r] [-v] file [files...]
```

Core Requirements

1. The script must:

- Parse the `file` command output to detect compression type
- Automatically choose the appropriate decompression method
- Place unpacked files in the same directory as the original archive
- Keep original archives intact
- Overwrite existing files automatically

2. Required compression formats:

- gunzip
- bunzip2
- unzip
- uncompress

Note: Design the script to make adding new compression formats simple.

Command Line Options

- `-v` (verbose):
  - Echo each file being decompressed and warn for each file that was NOT decompressed
- `-r` (recursive):
  - Traverse directories recursively and perform unpacking at each directory level

Important Behavior Rules

- **File detection:** Ignore extensions - use only the `file` command to determine type
- **Uncompressed files:** Take no action
- **Directory input:** Decompress all files in that directory (one level deep without -r)
- **Output:** Echo the amount of decompressed files (`Decompressed N archive(s)`)
- **Exit code:** Return the exact number of files NOT decompressed

Usage Examples

1. **Basic Usage**

Command: `$ unpack my-zip-file`
Output: `Decompressed 1 archive(s)`

- The contents of my-zip-file were extracted to its directory
- Exit code: 0 (success)

2. **Multiple Files**

Command: `$ unpack my-zip-file my-bz2-file`
Output: `Decompressed 2 archive(s)`

- Both archives were extracted to their respective directories
- Exit code: 0 (success)

3. **Non-Archive File**

Command: `$ unpack some-text-file`
Output: `Decompressed 0 archive(s)`

- No decompression performed (not an archive)
- Exit code: 1 (one file failed)

4. **Verbose Mode with Multiple Files**

   Command: `$ unpack -v *`
   Output:
   ```
   Unpacking my-bz2-file...
   Unpacking my-zip-file...
   Ignoring some-text-file
   Decompressed 2 archive(s)
   ```

   - Processed 3 files: 2 archives extracted, 1 ignored
   - Exit code: 1 (one non-archive file)

5. **Directory Handling (Non-Recursive)**

   Command: `$ unpack -v some-folder`
   Output:
   ```
   Unpacking a.zip...
   Unpacking b.bz2...
   Decompressed 2 archive(s)
   ```

   - Processed files in some-folder (one level deep)
   - Each archive extracted to its location within some-folder
   - Subfolders were ignored (no -r flag)
   - Exit code: 0 (all files were archives)

6. **Recursive Directory Handling**

   Command: `$ unpack -r some-folder`
   Output: `Decompressed 17 archive(s)`

   - Processed some-folder and all subfolders
   - Each archive extracted to its respective location
   - Found 20 total files: 17 archives extracted, 3 non-archives
   - Exit code: 3 (three files weren't archives)

Important Notes

1. Match the output format **exactly** as shown in examples
2. Follow good coding practices and write efficient code
3. Each directory should be processed appropriately based on the files it contains

Submission

- Submit via the link in your exam email
- Filename must be exactly: `unpack.sh`
- Use only English letters, numbers, and standard symbols
- Test thoroughly in a Linux environment before submitting - you only have 1 chance!

Good Luck!