CHAPTER 12 : BASH FOR YOUR SYSTEM

BY HANSIKA BHAMBHANEY

OBJECTIVES:

1.Download and obtain the latest version of bash.

2.Properly unpack and configure the bash source archive.

3.Compile and install bash from source code.

4.Identify and troubleshoot common installation issues.

5.Test the shell post-installation.

6.Install bash as your default login shell.

7.Locate helpful documentation and ask for support or report bugs.

12.1. Obtaining bash

This section of the chapter explains **how to get the bash shell** for your system. Whether you're a new user or a system administrator, the first step is obtaining the correct version of bash.

Where Can You Get bash?

You can **download bash** from official and trusted sources like:

GNU Project (Official Source)

- The **Free Software Foundation (FSF)** maintains the official version of bash.
- The GNU project's bash page provides:
  - Source code
  - Precompiled binaries (sometimes)
  - Documentation

We typically use tools like:

Using `wget` (command-line download tool)

```
wget https://ftp.gnu.org/gnu/bash/bash-5.2.tar.gz
```

Using `curl` (alternative to wget)

```
curl -O https://ftp.gnu.org/gnu/bash/bash-5.2.tar.gz
```

This downloads a compressed **source archive** of bash (a `.tar.gz` file).

- Always **download the latest stable version** unless you have a reason to use a specific one.
- Verify the **version compatibility** with your OS.
- You may also use **Linux distribution package managers**, such as:
    - `apt install bash` (Debian/Ubuntu)
    - `yum install bash` (RHEL/CentOS)
    - `brew install bash` (macOS using Homebrew)

However, if we want to **compile from source**, downloading from the GNU website is the preferred method.

12.2. Unpacking the Archive

After you've downloaded `bash` (e.g., `bash-5.2.tar.gz`), you need to **unpack** it so you can access the source code and build it.

Step 1: Unpack the `.tar.gz` File

Use the `tar` command:

```
tar -xvzf bash-5.2.tar.gz
```

`-x` → Extract the archive

`-v` → Verbose (shows files being unpacked)

`-z` → Uncompress gzip format

`-f` → Specifies the file

This creates a new folder like `bash-5.2/` with all the source files inside.

12.3. What's in the Archive

After extracting, you'll see several subdirectories and files inside the `bash-5.2/` directory.

12.3.1. Documentation

Contains:

- `README` – overview of the build process
- `INSTALL` – detailed install instructions
- `CHANGES` – list of changes from previous versions

12.3.2. Configuring and Building bash

Step 1: Navigate into the folder

```
cd bash-5.2
```

Step 2: Configure the source for your system

```
./configure
```

This checks your system and sets up the build environment.

Step 3: Build the shell

```
make
```

This compiles the bash source code into a binary executable.

12.3.3. Potential Problems

Some common issues:

- Missing dependencies (`make`, `gcc`, etc.)
- Incorrect system paths
- Permission errors (may need `sudo` for installation)

If `configure` or `make` fails, read the `README` or `INSTALL` files.

12.3.4. Installing bash as a Login Shell

Install it system-wide:

```
sudo make install
```

This usually installs bash to `/usr/local/bin/bash` or similar.

Set bash as your default login shell:

```
chsh -s /usr/local/bin/bash
```

We'll need to **log out and log back in** to apply the change.

12.3.6. Examples

Here's a full example from start to finish:

```
wget https://ftp.gnu.org/gnu/bash/bash-5.2.tar.gz
tar -xvzf bash-5.2.tar.gz
cd bash-5.2
./configure
make
make tests      # optional
sudo make install
chsh -s /usr/local/bin/bash
```

12-1. Configurable features

Feature	Description
alias	Support for aliases.
arith-for-command	Support for the alternate form of the `for' command that behaves like the C language *for* statement.
array-variables	Support for one dimensional arrays.
bang-history	C-shell-like history expansion and editing.
brace-expansion	Brace expansion.
command-timing	Support for the **time** command.
cond-command	Support for the **[[** conditional command.

Feature	Description
cond-regexp	Support for matching POSIX regular expressions using the =~ binary operator in the **[[** conditional command.
directory-stack	Support for the **pushd**, **popd**, and **dirs** directory manipulation commands .
disabled-builtins	Whether a built-in can be run with the **builtin** command, even if it has been disabled with **enable -n**.
dparen-arithmetic	Support for **((...))** .
help-builtin	Support for the **help** built-in.
history	History via the **fc** and **history** commands .

Feature	Description
progcomp	Programmable completion facilities. If *readline* is not enabled, this option has no effect .
readline	*readline* editing and history capabilities.
restricted	Support for the restricted shell, the **-r** option to the shell, and **rbash**.
select	The **select** construct.
usg-echo-default xpg-echo-default	Make **echo** expand backslash-escaped characters by default, without requiring the **-e** option. This sets the default value of the *xpg_echo* shell option to *on*, which makes the *bash* **echo** behave more like the version specified

Feature	Description
job-control	Job control via **fg**, **bg**, and **jobs** if supported by the operating system.
multibyte	Support for multibyte characters if the operating system provides the necessary support.
net-redirections	Special handling of filenames of the form */dev/tcp/HOST/PORT* and */dev/udp/HOST/PORT* when used in redirections.
process-substitution	Whether process substitution occurs, if supported by the operating system.
prompt-string-decoding	Whether backslash escaped characters in PS1, PS2, PS3, and PS4 are allowed .

Examples

The bash archive also includes an examples directory. This directory contains some subdirectories for scripts, functions, and examples of startup files. The startup files in the startup-files directory provide many examples of what you can put in your own startup files. In particular, bash_aliases gives many useful aliases. Bear in mind that if you copy these files wholesale, you'll have to edit them for your system because many of the paths will be different. The functions directory contains about 50 files with function definitions that you might find useful. Among them are:
1}basename The basename utility, missing from some systems

2}dirfuncs Directory manipulation facilities

3}dirname The dirname utility, missing from some systems

12.4 Reporting Bugs

What to Include in Your Bug Report:

1. **Operating System Details:**
   - Mention the **name and version** of your OS.
   - Example:
     `Ubuntu 22.04 LTS (64-bit)` or `macOS 13.5 Ventura`
2. **Bash Version:**
   - Run this command in terminal:

     ```
     echo $BASH_VERSION
     ```
3. Include the full output.
   Example: `5.2.15(1)-release`

Exact Commands That Caused the Issue:

- Write the **exact command(s)** you typed that triggered the bug.
- Copy-paste directly from your terminal if possible.

Expected Output:

- Clearly explain what you **expected to happen**.
- Example: “Expected the script to print ‘Hello World’ and exit.”

Actual Output:

- Show what **actually happened**.
- Include error messages, incorrect behavior, or nothing if it failed silently.

Environment Details (Optional but helpful):

- Were you in an SSH session, terminal emulator, or script?
- Any environment variables or special `.bashrc` settings?

Best Practices & Tips Before Reporting:

1. **Try to Reproduce the Bug:**
    - Re-run the same steps and confirm it's not a one-time issue.
    - Try on a fresh terminal or new user account if needed.
2. **Test on the Latest Version:**
    - Ensure you’re using the most recent bash version.
    - If not, try updating before reporting the bug.

3. **Search Existing Reports:**
    - Check the mailing list archive or forums to avoid duplicates:
        - https://lists.gnu.org/archive/html/bug-bash/
        - Stack Overflow, Reddit, or UNIX StackExchange
4. **Be Clear and Respectful:**
    - Write a **short and clear subject** line.
    - Keep the message polite and professional.
    - Use bullet points or numbered steps to improve readability.

Sample Bug Report Template:

```
Subject: Bug in bash 5.2.15 - Unexpected behavior when using 'read' with here-strings

OS: Ubuntu 22.04 LTS (64-bit)
Bash Version: 5.2.15(1)-release

Command:
read var <<< "hello"

Expected:
Variable 'var' should contain the string "hello"

Actual:
Variable 'var' is empty

Additional Info:
Occurs only in interactive mode. Works fine in scripts.
```

CONCLUSION

Chapter 12 equips users with the practical knowledge to obtain, compile, and install the Bash shell from source, ensuring they can work with the latest version regardless of their system's default shell. It walks through downloading bash from official GNU sources, unpacking and building it using standard tools (`configure`, `make`, `make install`), and setting it as the default login shell. The chapter also emphasizes the importance of proper testing, addresses common installation issues, and provides guidance on how and where to seek help or report bugs. Altogether, it empowers users to confidently manage their own Bash installation and contribute to its community when needed.