Managing the Environment

This handout covers key concepts related to managing the shell environment, including environment variables, the `PATH` variable, persistent configurations, passing environments to programs, and creating aliases.

---

1. Introduction to the Shell Environment

The shell environment enables customization of your command-line interface and provides configuration settings for running programs. It includes:

- Environment Variables
- Persistent Configuration Files (e.g., `.bashrc`)
- Command Aliases

---

2. Environment Variables

Purpose

Environment variables store configuration settings that influence shell behavior and programs.

- Standard convention: uppercase letters (e.g., `HOME`, `PATH`).

Common Commands

- **View all environment variables:**

```
env
```

- **Access an environment variable:**

Recommended syntax:

```
echo "${HOME}"
```

> ***Avoid omitting quotes or curly braces*** *for consistency and correctness.*

- **Set a new environment variable (temporarily):**

```
export MY_VARIABLE="some value"
```

> *Temporary environment variables affect only the current shell session.*

Example usage (changing `HOME` temporarily)

```
export HOME="/home"
cd        # Changes directory based on new HOME value
```

---

3. The PATH Variable

`PATH` is a special environment variable containing directories that Linux searches when you type a command.

- Example content of PATH:

```
/usr/local/bin:/usr/bin:/bin
```

Important concepts

- Directories in `PATH` are searched from left to right.
- If two programs share the same name, the first one found is executed.

Modifying PATH temporarily

```
export PATH="${PATH}:/home/yourname/bin"
```

- Recommended: **Keep system directories at the beginning** to avoid overriding essential commands.
- Clean up `PATH` regularly to avoid obsolete entries.

---

4. Persisting Configuration (`.bashrc`)

To persist environment variable changes across shell sessions, store them in the `.bashrc` file in your home directory (`~/.bashrc`).

Example

Edit `.bashrc` :

```
nano ~/.bashrc
```

Add changes at the bottom:

```
export PATH="${PATH}:/home/yourname/bin"
```

- Save (`Ctrl+O`) and exit (`Ctrl+X`).
- Restart terminal or run:

```
source ~/.bashrc
```

Why `.bashrc` ?

`.bashrc` is loaded by interactive, non-login shells (typical terminal sessions). On most systems (like Ubuntu), `.bashrc` is also sourced from `.profile` , making it the best choice for persistent user-specific configurations.

---

5. Passing Environment Variables to Programs

Child processes inherit environment variables from their parent shell.

Setting variables inline (only for one command)

```
MY_VARIABLE="database.local" python3
```

In Python, accessing the variable:

```
import os
print(os.getenv("MY_VARIABLE"))
```

Exporting environment variables (whole session)

```
export MY_VARIABLE="database.local"
```

- All subsequent programs inherit this variable.

Bash variables (not environment variables)

Bash variables without `export` aren't passed to child processes:

```
MY_VARIABLE="local"  # Only accessible within this bash session
```

---

6. Creating Aliases

Aliases simplify frequently used commands or command sequences.

Creating an alias

Syntax:

```
alias alias_name="command_to_execute"
```

Example—create alias for navigating to desktop:

```
alias go_desktop="cd ~/Desktop"
```

- Usage:

```
go_desktop
```

Listing aliases

```
alias
```

Combining multiple commands

```
alias update_sys="sudo apt update && sudo apt upgrade"
```

- This runs both commands if the first succeeds.

Removing an alias

```
unalias go_desktop
```

Making aliases permanent

Add alias commands to `~/.bashrc` to persist them across sessions:

```
nano ~/.bashrc
```

At the end add:

```
alias go_desktop="cd ~/Desktop"
```

Apply changes immediately with:

```
source ~/.bashrc
```