Directory Navigation

1. `pwd`
   Role: Prints the current working directory.
2. `ls`
   Role: Lists the files and directories in the current working directory.
3. `ls Desktop/`
   Role: Lists the contents of the `Desktop` directory.
4. `ls -l`
   Role: Displays the detailed list of files and directories in long format.
5. `ls -h`
   Role: Lists files with human-readable sizes.
6. `ls -lh`
   Role: Combines long format (`-l`) and human-readable sizes (`-h`).
7. `ls -a`
   Role: Lists all files, including hidden ones (those starting with a `.`).
8. `ls -lha`
   Role: Combines long format (`-l`), human-readable sizes (`-h`), and includes hidden files (`-a`).
9. `cd Do`
   Role: Starts navigating to a directory matching the prefix `Do`.
10. `cd Desktop/`
    Role: Changes the working directory to `Desktop`.
11. `cd ~/Desktop/qwe/zxc`
    Role: Changes to the `zxc` directory inside `qwe` on `Desktop`.
12. `cd -`
    Role: Moves to the previous working directory.
13. `cd ..`
    Role: Moves one level up in the directory hierarchy.
14. `cd`
    Role: Returns to the home directory.

Directory Operations

15. `mkdir ~/Desktop/asd`
    Role: Creates a directory named `asd` on the `Desktop`.
    Error: Directory already exists.
16. `mkdir ~/Desktop/qwe`
    Role: Creates a directory named `qwe` on the `Desktop`.
17. `mkdir ~/Desktop/qwe/zxc/asd`
    Role: Creates the nested directory structure under `qwe/zxc/asd`.
    Error: Parent directories don’t exist.

18. `mkdir -p ~/Desktop/qwe/zxc/asd`
    Role: Creates the directory structure, creating parent directories as needed.
19. `rmdir ~/Desktop/qwe/zxc/`
    Role: Removes an empty directory named `zxc`.
    Error: Directory is not empty.
20. `rmdir ~/Desktop/qwe/zxc/asd/`
    Role: Removes the `asd` directory since it's empty.
21. `rm -r 'Hello worl'`
    Role: Recursively deletes the `Hello worl` directory.

File Operations

22. `touch Hello`
    Role: Creates an empty file named `Hello` in the current directory.
23. `touch ~/Desktop/Hello`
    Role: Creates an empty file named `Hello` on the `Desktop`.
24. `touch ~/Desktop/"Hello world"`
    Role: Creates an empty file named `Hello world` on the `Desktop`.
25. `echo "Hello world" > Hello`
    Role: Writes `Hello world` into the file `Hello`, overwriting its content.
26. `echo "Hello world" >> Hello`
    Role: Appends `Hello world` to the file `Hello`.
27. `cat Hello`
    Role: Displays the content of the file `Hello`.
28. `cp qaz qwe`
    Role: Copies the file `qaz` to the directory `qwe`.
29. `cp -r ~/Desktop/main ~/Desktop/aux ~/Desktop/qwe/`
    Role: Copies the files `main` and `aux` into the `qwe` directory.
30. `mv ~/Desktop/qwe/main ~/Desktop/qwe/last`
    Role: Renames or moves `main` to `last` in the `qwe` directory.
31. `mv qaz first`
    Role: Renames or moves the file `qaz` to `first`.

File Compilation and Execution

32. `gcc file.c -o out`
    Role: Compiles the C program in `file.c` and creates an executable named `out`.
33. `./out`
    Role: Executes the compiled C program `out`.

Process Management

34. `ps`
    Role: Lists the currently running processes in the terminal session.

Manual and Editor

35. `nano`
    Role: Opens the `nano` text editor.
36. `nano qaz`
    Role: Opens the file `qaz` in `nano` for editing.
37. `man`
    Role: Opens the manual interface, prompting for a command to display.
38. `man ls`
    Role: Displays the manual page for the `ls` command.

Additional commands

`find`

- **Role:** Searches for files and directories based on criteria.
- Example: `find ~/Desktop -name "*.txt"` (Finds all `.txt` files on the Desktop.)

`grep`

- **Role:** Searches for text patterns in files.
- Example: `grep "error" logfile.txt`

C programming

Here's an explanation of each command and its role, based on your terminal session:

Directory Navigation

1. **`cd Desktop/`**
    - **Role:** Changes the current directory to `Desktop`.
2. **`~/Desktop`**
    - **Role:** Refers to the full path of the `Desktop` directory in the home folder.

File Compilation and Execution

3. **`gcc file.c -o out`**
    - **Role:** Compiles the C program in `file.c` into an executable named `out`.
4. **`./out`**
    - **Role:** Executes the compiled program `out` located in the current directory (`./` ensures the current path is included).
5. **`~/Desktop/out`**
    - **Role:** Executes the `out` file using its absolute path.
6. **`out`**
    - **Role:** Fails to execute because `out` is not in the system's `PATH`.

File Content and Redirection

7. **`echo 7 > circle.txt`**
    - **Role:** Writes the number `7` to the file `circle.txt`, creating or overwriting it.
8. **`cat circle.txt`**
    - **Role:** Displays the contents of `circle.txt`.
9. **`cat output.txt`**
    - **Role:** Displays the contents of `output.txt`.

Error and Input Handling in Programs

10. **`./out` (multiple runs with varying outputs)**
    - **Role:** Runs the program multiple times with different conditions. Outputs vary depending on program logic (e.g., missing input files, command-line arguments).
11. **`./out Hello`**
    - **Role:** Runs the program with the word `Hello` as input, demonstrating how arguments are processed.
12. **`./out 1 2 3`**
    - **Role:** Runs the program with three numeric arguments to demonstrate command-line argument handling.

Process Management

13. **Fork and Process Messages**
    - **Role:** Demonstrates the program creating child processes using `fork()` in the code. Messages show process IDs (PIDs) and parent-child relationships.

File and Directory Listing

14. **`ls`**
    - **Role:** Lists files and directories in the current directory.
15. **`/bin/ls`**
    - **Role:** Explicitly runs the `ls` command from its binary location in `/bin`.

Output Redirection

16. **File Listing in Program Output**
    - **Role:** The program dynamically lists directory contents (similar to `ls`) as part of its logic.

Advanced Process Demonstration

17. **`./last`**
    - **Role:** Runs another compiled program (`last`) that uses `fork()` and `exec()` to demonstrate process management and execution of other commands.