



Fish Shell Basics

Core Syntax

Variable Assignment	<code>set variable_name value</code>
Accessing Variables	<code>\$variable_name</code> or <code>\${variable_name}</code>
String Concatenation	<code>set combined "\$var1 \$var2"</code>
Command Substitution	<code>set output (command)</code>
Comments	<code># This is a comment</code>
Exit Status	<code>echo \$status</code> (after command execution)

Control Flow

If Statement	<pre>if test condition # commands else if test condition # commands else # commands end</pre>
For Loop	<pre>for i in item1 item2 item3 # commands using \$i end</pre>
While Loop	<pre>while test condition # commands end</pre>
Switch Statement	<pre>switch \$variable case value1 # commands case value2 # commands case * # default commands end</pre>

Functions

Function Definition	<pre>function function_name # commands end</pre>
Calling Functions	<code>function_name arg1 arg2</code>
Function Arguments	<code>\$argv[1]</code> , <code>\$argv[2]</code> , etc.
Return Values	Use <code>return value</code> (sets <code>\$status</code> variable)

Common Commands & Utilities

File and Directory Operations

Create Directory	<code>mkdir directory_name</code>
Remove Directory	<code>rmdir directory_name</code> (empty directory) <code>rm -r directory_name</code> (recursive delete)
Create File	<code>touch file_name</code>
Remove File	<code>rm file_name</code>
Copy File	<code>cp source_file destination_file</code>
Move/Rename File	<code>mv source_file destination_file</code>
List Files	<code>ls</code>

Text Manipulation

Print to Console	<code>echo message</code>
Grep (Search)	<code>grep pattern file_name</code>
Sed (Stream Editor)	<code>sed 's/old/new/g' file_name</code>
Awk (Pattern Scanning)	<code>awk '{print \$1}' file_name</code>
String Length	<code>string length \$variable</code>

Process Management

Run Command in Background	<code>command &</code>
List Running Processes	<code>jobs</code> or <code>ps</code>
Kill Process	<code>kill process_id</code>
Foreground Process	<code>fg job_id</code>

Advanced Fish Scripting

Error Handling

Check Exit Status	<code>if test \$status -ne 0</code>
Try...Catch (Simulated)	<code>command echo "Error occurred"</code>
Using <code>command</code> to ignore functions	Forces execution of external command instead of shell function with same name. Useful in scripts.

Signals

Trapping Signals	Fish does not directly support <code>trap</code> . Use external tools or workarounds.
------------------	---

Input and Output Redirection

Redirect Output to File	<code>command > file.txt</code> (overwrite) <code>command >> file.txt</code> (append)
Redirect Input from File	<code>command < file.txt</code>
Pipe Output	<code>command1 command2</code>
Redirect Standard Error	<code>command ^ file.txt</code>

Working with Lists

Creating a List	<code>set my_list item1 item2 item3</code>
Accessing List Elements	<code>echo \$my_list[1]</code> (first element)
List Length	<code>count \$my_list</code>
Iterating Over a List	<pre>for item in \$my_list echo \$item end</pre>

Configuration and Customization

Configuration Files

Global Configuration	<code>~/.config/fish/config.fish</code>
Functions Directory	<code>~/.config/fish/functions/</code> (for custom functions)
Autocompletions Directory	<code>~/.config/fish/completions/</code> (for custom autocompletions)

Custom Functions

Creating a Function	<pre>function greet echo "Hello, \$argv[1]!" end</pre>
Making it Persistent	Save the function in <code>~/.config/fish/functions/greet.fish</code>

Autocompletions

Creating Autocompletions	Use <code>complete</code> command to define autocompletions. Example: <code>complete -c mycommand -f -a "option1 option2"</code>
Custom Completion Files	Store completion definitions in <code>~/.config/fish/completions/mycommand.fish</code>

Aliases

Creating Aliases (Abbreviations)	<code>abbr -la 'ls -la'</code>
Making Aliases Persistent	Add <code>abbr</code> commands to <code>~/.config/fish/config.fish</code>