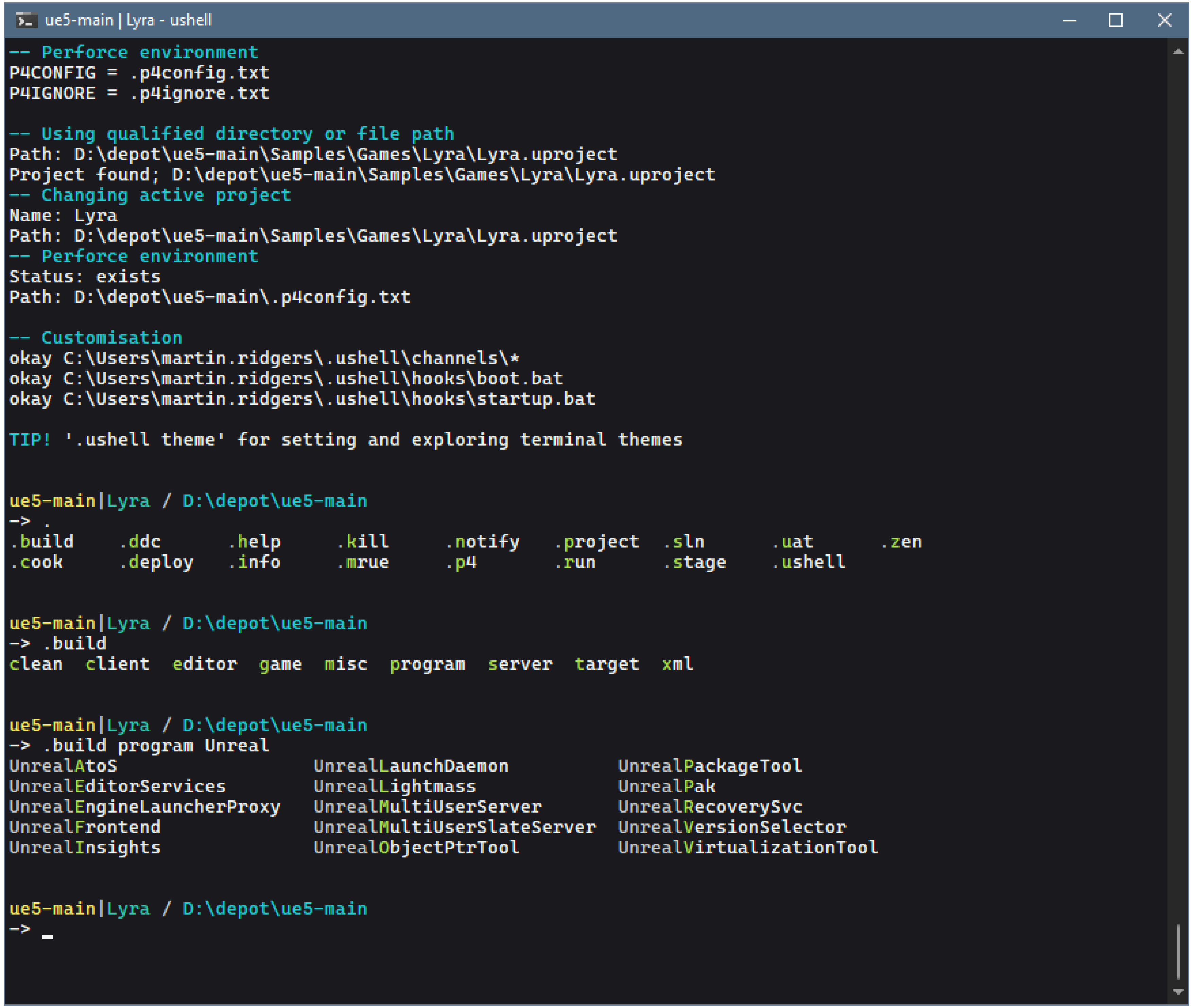
Ushell

Explore ways to use the ushell command line interface in Unreal Engine.

The **ushell** feature is a command line interface for working with Unreal Engine projects.



```
-- Perforce environment
P4CONFIG = .p4config.txt
P4IGNORE = .p4ignore.txt

-- Using qualified directory or file path
Path: D:\depot\ue5-main\Samples\Games\Lyra\Lyra.uproject
Project found; D:\depot\ue5-main\Samples\Games\Lyra\Lyra.uproject
-- Changing active project
Name: Lyra
Path: D:\depot\ue5-main\Samples\Games\Lyra\Lyra.uproject
-- Perforce environment
Status: exists
Path: D:\depot\ue5-main\.p4config.txt

-- Customisation
okay C:\Users\martin.ridgers\.ushell\channels\*
okay C:\Users\martin.ridgers\.ushell\hooks\boot.bat
okay C:\Users\martin.ridgers\.ushell\hooks\startup.bat

TIP! '.ushell theme' for setting and exploring terminal themes


ue5-main|Lyra / D:\depot\ue5-main
-> .
.build    .ddc      .help     .kill     .notify   .project  .sln      .uat      .zen
.cook     .deploy   .info     .mrue     .p4       .run      .stage    .ushell


ue5-main|Lyra / D:\depot\ue5-main
-> .build
clean  client  editor  game  misc  program  server  target  xml


ue5-main|Lyra / D:\depot\ue5-main
-> .build program Unreal
UnrealAtoS                   UnrealLaunchDaemon           UnrealPackageTool
UnrealEditorServices         UnrealLightmass              UnrealPak
UnrealEngineLauncherProxy    UnrealMultiUserServer        UnrealRecoverySvc
UnrealFrontend               UnrealMultiUserSlateServer   UnrealVersionSelector
UnrealInsights               UnrealObjectPtrTool          UnrealVirtualizationTool


ue5-main|Lyra / D:\depot\ue5-main
->
```

It features commands for many foundational UE operations, including:

- Building code
- Cooking and staging data
- Running titles
- Executing commands
- Running trace sessions for Unreal Insights
- And more

Ushell supports rich tab completion, includes detailed inline documentation, and has a persistent searchable command history. This page provides information on how to quickly get started using ushell.

How to Run ushell

To start ushell, open your engine's install directory (for example, `C:/EpicGames/UE_5.4`) and run the following batch file:

Relative path in engine install directory

```
Engine/Extras/ushell/ushell.bat
```

Copy full snippet

> Launcher builds of UE 5.4 do not include ushell, but source code builds do. This issue is corrected in UE 5.5.

Every command starts with a dot/period character. For example, the `.build` command will build projects in various configurations. You can use tab completion to aid in discovery of commands and arguments.

Help and Documentation

View a list of all available commands by typing the `.help` command, and access each command's usage guide by specifying the `--help` argument:

ushell commands

```
.help
.build editor --help
```

Copy full snippet

More detailed documentation can be found in ushell's readme file, covering topics such as extending ushell with custom commands, scripting with ushell, running on POSIX-based platforms, and integrating with other shells. For convenience, this can be accessed using `.help readme`.

Example Commands

To help understand the use cases for ushell, the following is a non-exhaustive list of command examples you might use:

Example Command	Description
`.build editor`	Build the editor for the active project.
`.build game win64`	Build the game runtime for the active project for the Win64 platform.
`.build program UnreralInsights shipping`	Build Unreal Insights in shipping configuration.

Example Command	Description
`.cook game win64`	Cook data for the game runtime for the Win64 platform.
`.stage game mac`	Stage locally built game/client executables and previously cooked data for mac.
`.run editor`	Run the editor for the active project.
`.run game Win64 --trace -- -ExecCmds="TrySurfing"`	Launch the locally built binaries for your active project for the Win64 platform. Send an Unreal Insights trace session to the development host, and execute the "TrySurfing" command.
`.run program UnrealInsights shipping`	Run Unreal Insights in a shipping configuration.
`.sln generate`	Generate a Visual Studio solution for the active project.
`.info`	Display info about the current session.

ⓘ More detailed information is available for each of the above commands with the `.help` command or by using the `--help` argument.

Each of the ushell commands accepts a `--help` argument that will show documentation for the command, details on how it is invoked, and descriptions for the available options.

Tab Completion and Command History

Anyone familiar with editing commands and recalling previous commands in Bash (such as Readline) will feel at home in a ushell prompt.

Extensive context-sensitive tab completion is available for commands and their arguments. Pressing the Tab key helps to discover both commands and their arguments, and aids in quick and convenient command entry. For example:

Example Input	Description
`.<tab><tab>`	Displays available commands.
`.b<tab>`	Completes `.build`.
`.run <tab><tab>`	Shows options for the `.run` first argument.
`.build editor --p<tab>`	Adds `--platform=`. Further tabs complete platforms.

Ushell maintains a history of previously run commands that persists from one session to the next. Previous commands can be conveniently recalled in a few ways.

To step backward through prior commands by prefix, use PgUp:

Example Input	Description
`.bu<pgup>`	Cycle back through commands that started with "`.bu`"
` .run game xb`	Iterate through previous runs on Xbox.

You can perform a more thorough search through your incremental history by pressing `Ctrl-R`. This displays a prompt to enter a search string, and will display the latest command with a match. Pressing `Ctrl-R` again will step backward through commands that match the search string. If you press `Ctrl-S`, it will step forward instead.