That's an excellent way to integrate the learning and progression directly into the core gameplay loop, creating a true "hack-to-learn" system. Command and knowledge acquisition becomes a resource in itself.
Here is the design for the **Command and Knowledge Acquisition System** in **Code Nexus**.

Command and Knowledge Acquisition System 🧠

The system is designed to gate access to the man pages and the core execution environment until the command is *discovered* and *validated* by the player.

A. The "Locked" State

Initially, the **NexusScript** commands listed in the database (e.g., pivot, thread spawn, raw) are **unrecognized** by the player's Virtual Computer (VC).

Action	Result	Learning Incentive
Player attempts to use pivot	**NexusShell** returns: "Error: Command pivot not found. Consult your internal **K-Map**."	**Incentive:** The player must actively find the knowledge required to enable the command.
Player attempts man pivot	**NexusShell** returns: "Error: Manual Page not available for pivot. Knowledge State: **LOCKED**."	**Incentive:** The in-game help is useless until the player earns the knowledge.

B. Command Discovery Mechanisms

Players gain access to new commands through a variety of engaging in-game activities:

1. **Mission Rewards (Guided Learning):**
    - **Mechanism:** Completing a mission that *requires* the concept of pivoting (e.g., accessing an internal network) grants the player a **Knowledge File** called pivot_v1_concept.kns.
    - **Validation:** Running the file (cat pivot_v1_concept.kns) unlocks the pivot command and its initial man page entry.
2. **Filesystem Harvesting (Exploration):**
    - **Mechanism:** Successfully exploiting a target server (PvE or PvP) often reveals a "hidden" **Configuration File** (.cfg) or **Documentation File** (.doc) on the remote system.
    - **Validation:** The player must **cat** or **copy** the remote file to their local VC. The VC's kernel scans the file's contents for unrecognized syntax. If the file contains the text structure CMD_DECLARE: raw_send, the command raw is unlocked.
3. **PvP/Player-Crafted Modules (Social/Advanced):**
    - **Mechanism:** A high-level opponent's defense script, when captured and analyzed, may contain code the player hasn't unlocked yet (e.g., the thread spawn command).
    - **Validation:** The player must successfully **run --debug** or **edit** the captured script. The kernel recognizes the new command syntax and asks, "New command structure found: **thread spawn**. Do you wish to integrate this into your K-Map?"

C. The Knowledge-Map (K-Map) Integration

The **Knowledge-Map (K-Map)** is the central repository for learned commands and concepts.

- **Syntax:** kmap view or kmap view cmd [command]
- **Function:** Once a command is validated through discovery, it moves from the "Locked" state to the "Integrated" state in the K-Map. The man page becomes accessible, and the VC kernel accepts the command as valid input.

K-Map State	Example Command	Player Capability
LOCKED	raw	Shell returns "Command not found."
DISCOVERED	raw	Player has found a .doc file mentioning raw.
INTEGRATED	raw	Player has run the .doc file, the man raw page is available, and the command is executable.

D. Advanced Knowledge Acquisition (Script Creation)

For the most complex and specialized commands (e.g., the high-level functions of the **Exploit_Object**), the player is required to **synthesize** the knowledge.

- **Challenge:** The player receives a fragmented document or multiple files that contain the *components* of a new command or object method, but not the whole thing.
- **Goal:** The player must **write a new NexusScript module** that correctly chains these fragmented concepts together.
- **Validation:** When the player runs the custom module (run my_new_exploit), the VC kernel checks the *logic* of the new script. If the script successfully implements the core concept (e.g., memory allocation followed by a payload injection), the kernel triggers the unlock:
    - "Synthesis Complete! New object method **$exploit.buffer_size** unlocked. K-Map updated."

This forces players to move beyond simple copying and pasting, requiring genuine understanding and creative application of the concepts to unlock the full potential of the **NexusScript** language.