

KAIST | FALL 2025

# KERNEL SYSTEM SECURITY

*“new components for new operating systems”*

—

*AI-powered-Shell*

Arthur Bidet – GSIS

# PROJECT PRESENTATION

## SUBJECT

**WHICH FUNCTIONALITY OR  
MODULE DOES THE OS/  
KERNEL NEED?**

## RULES

**EXPLAIN THE ASSOCIATED  
BACKGROUND AND YOUR  
REASON WHY SUCH  
FUNCTIONALITY WILL BE  
NEEDED IN THE FUTURE**

## TABLE OF CONTENTS

1. Problem statement	4
2. Related work	5
3. Solution implementation	6
4. Assessment and conclusion	8

## 2. RESEARCH QUESTION

### BASE OBSERVATION

- As the LLM usage are increasing, no real progress has been seen to enhance user experience since the first versions.
- Switching between windows and interruptions in tasks creates an overhead for the user and reduces productivity.

## 3. RELATED WORK

Google ai in bash github

AI Overview

There are numerous GitHub projects that integrate AI capabilities into the Bash command line interface. These tools primarily use large language models (LLMs), like OpenAI's GPT models or local models, to generate, suggest, or execute commands from natural language prompts.

Here are some notable AI in Bash GitHub projects:

- [BuilderIO/ai-shell](#): A popular CLI tool (installed via npm) that converts natural language prompts into shell commands. It allows users to confirm, revise, or cancel the suggested command before execution.
- [TheR1D/shell\\_gpt](#): Known as ShellGPT, this robust command-line productivity tool generates shell commands, code snippets, and documentation from AI models. It is compatible with Bash, Zsh, PowerShell, and other major shells.
- [backus/ai.sh](#): A pure Bash script that acts as a "Copilot for the terminal". It uses the `openai` CLI to generate commands, offering a simple interactive UI (using `skim`) to choose whether to run, copy, or discard the output.
- [M4R14/bash-ai](#): A tool that provides intelligent suggestions and automation for bash commands using OpenAI's API.
- [davigegat/LAIB-Local-AI-Bash](#): This project focuses on privacy and flexibility by using local LLMs (through LMStudio) to generate commands within an AI-powered terminal interface built with Python and Tkinter.
- [IBM/clai](#): Command Line Artificial Intelligence (CLAI) is a framework that allows AI agents to be integrated into the command line environment, assisting with various tasks.

These tools generally require you to set up an API key for services like OpenAI or Anthropic, or configure a local LLM environment, to function.

Dive deeper in AI Mode

davigegat/LAIB-Local-AI-Bash - GitHub  
AI-powered Bash terminal built with Python, Tkinter, tkterm, using local LLM through LMStudio for natural language...

M4R14/bash-ai - GitHub  
Bash AI is a command-line tool that uses the power of OpenAI's GPT-3 to provide intelligent...

TheR1D/shell\_gpt - GitHub  
ShellGPT. A command-line productivity tool powered by AI large language models (LLM)....

Show all

## KEY POINTS

- Get the **context**;
- Connect shell to **LLM API**;
- Send **sensitive content** to service providers;
- Send sensitive data to ... USA, China ? This could make my European ears bleed.

## 4. SOLUTION IMPLEMENTATION

### AI-POWERED SHELL ASSISTANT

Loading context from: `/tmp/ai_powered_shell_arthub.json`

Question: The command 'why isnt it working' was not found. What should I do?

- ✓ Detected OS: **Arch Linux**
- ✓ Package Manager: **pacman**
- ✓ Debug context saved to `/tmp/ollama_context_debug_12712.md`

... Calling Ollama API (model: **mistral**)...

### OLLAMA RESPONSE

**Diagnosis:** The user encountered an error while trying to execute the file `./ch64`, which is due to a mismatch in the executable format and the system architecture.

**Explanation:** The executable `ch64` is a 32-bit MIPS binary, but the user's system is running on an x86\_64 architecture. As a result, the system cannot execute this file.

#### Actionable Solutions:

1. Recompile the `ch64` executable for the correct architecture or use a cross-compiler to create a version compatible with x86\_64 systems.
2. If this is not an option, you can create a chroot environment with the appropriate MIPS system and run the binary within that environment. To do so, install `qemu-user-static` and run `sudo chroot . /usr/bin/qemu-mips-static ./ch64`.

#### Additional Context:

It might be helpful to know whether you have the necessary development tools installed for building MIPS binaries on your Arch Linux system. If not, consider installing the command like `sudo pacman -S mips-newlib`.

**Warning:** The chroot method may require elevated privileges and should be used with caution. Always ensure you understand the implications before executing such commands.



## 4. SOLUTION IMPLEMENTATION

```
archlinux% in which file is there the string "True" in this directory
```

```
Error: Command 'in' not found.
```

```
Would you like AI assistance? [Q]uick / [D]eep / [N]o (default: Q): D
```

```
Deep Model
```

```
... Step 1: Requesting for the workers using Ollama API (model: mistral)...
```

```
... Step 2: Running following workers: {'file_analysis': ['file1.txt', 'file2.txt', 'file3.txt', 'file4.txt', 'file5.txt']}...
```

```
[BEGIN USER PROMPT]
```

```
... Step 3: Index the request using Ollama API (model: mistral)...
```

### OLLAMA RESPONSE

In the given context, the user is trying to find a file containing the string "True" within the current directory `/home/arthur/Downloads/test_dir`. From the "Workers Results" section, it is clear that only `file3.txt` contains the string "True".

To solve this problem, you can run the following command:

```
command: find . -type f -exec grep -l "True" {} \;
```

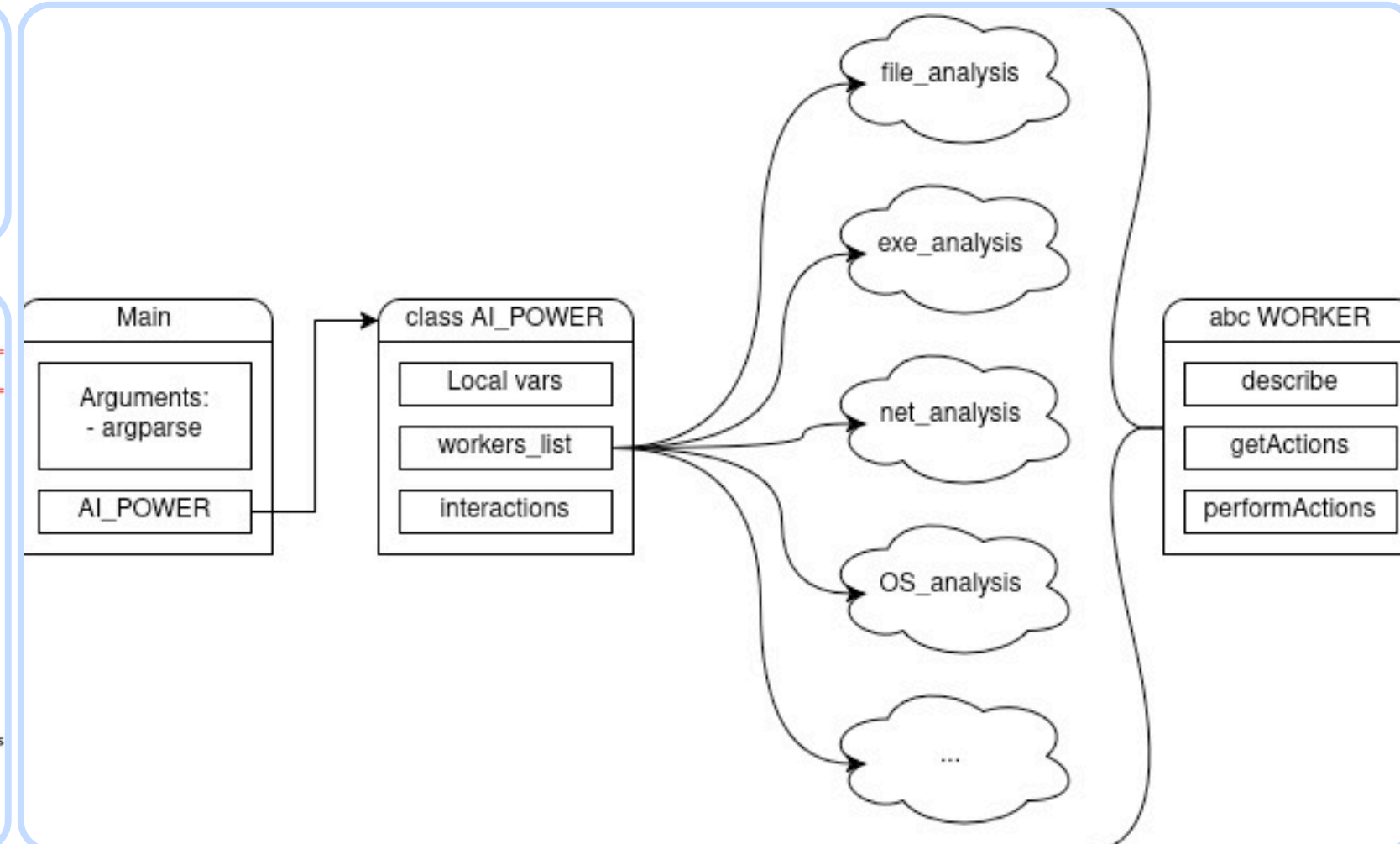
This command uses `find` to search for files recursively within the current directory and its subdirectories. The `-type f` option limits the search to files, not directories. The `-exec grep -l "True" {} \;` part executes the `grep` command on each file found by `find`, looking for the string "True", and prints only the names of the files that contain it (the `-l` option).

As a side note, if the user wants to limit the search only within the current directory, you can modify the command as follows:

```
command: grep -L "True" file1.txt file2.txt file3.txt file4.txt file5.txt
```

This command searches for the string "True" in each of the five files directly (without recursion), and prints only the names of the files that do not contain it (the `-L` option). In this case, since we know that only `file3.txt` contains the string "True", you can replace all the other file names with a wildcard like so:

```
command: grep -L "True" *
```



## 5. ASSESSMENT AND CONCLUSION

### VALIDATION PROCESS

- 1- Logging feature
- 2- Quality of Service

### SCENARIOS

- 1- Find the biggest file of the directory
- 2- Find the file containing the string "True"
- 3- Why cant I reach internet on firefox (DNS issue)
- 4- Wrong file type

### CONCLUSION - TO GO FURTHER

- Long delay for deep mode ( $\geq 2 \times$  classic mode);
  - Not always accurate;
- But
- Tests made with a GTX 1060 and very small model;
  - Still in an early phase, efficiency can be improved.

Some ideas...

- 1- Reduce the size of the prompts (currently very detailed);
- 2- Use better hardware;
- 3- Create new workers.



**THANK YOU FOR YOUR ATTENTION!**

Arthur Bidet – GSIS