成大 116 級資訊專題展

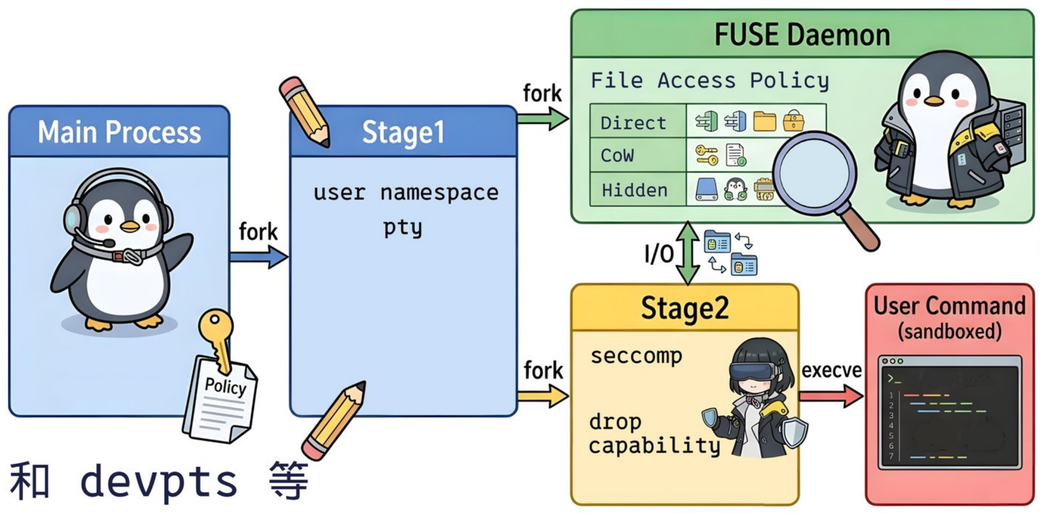
基於Linux Namespace和FUSE的沙箱

A process sandbox based on Linux namespace and FUSE

指導教授：莊坤達　成員：邱繼叡

1 系統架構

本系統由三個 process 組成。
Main process 負責處理 pty 和 signal
FUSE Daemon 負責代理沙箱和外部的互動。
最後是沙箱內部的程式，跑在全部獨立的 namespace。



啟動流程：
先讀取沙箱的 config，啟動背景服務掛載虛擬根目錄
FUSE daemon會進入 mount namespace 後，掛載 procfs 和 devpts 等
最後使用 seccomp 等機制，安全執行不可信任的程式

2 專題動機

隨著 AI 輔助開發(AI-Assisted Development) 的興起，開發者越來越傾向將任務委託給大型語言模型(LLM)。
然而，當 AI 代理(AI Agent) 有了實際執行的能力，安全性與隔離性便成為不可忽視的關鍵挑戰。
傳統的容器技術雖然能提供一定程度的隔離，但在存取檔案的細粒度上存在局限。

本專題提出一種基於 Linux Namespace 與 FUSE 的沙箱，
旨在為 AI 程式執行提供一個安全、可控的隔離環境。
透過結合操作 Linux 核心的命名空間和 FUSE，
我們希望在不妨礙 AI 存取必要檔案的前提下，
防止檔案相關的惡意行為。

方案	缺點
run in docker	opencode 有 bug(#14194)
devcontainer	需要維護額外環境
opencode內建機制	只防止 agent 跑 bash command 時直接存取外部檔案

3 核心技術

filesystem in userspace 提供完整檔案系統控制
利用 seccomp 限制系統呼叫
利用 linux namespace 實現 rootless container

使用 unix socket
同步多個實例

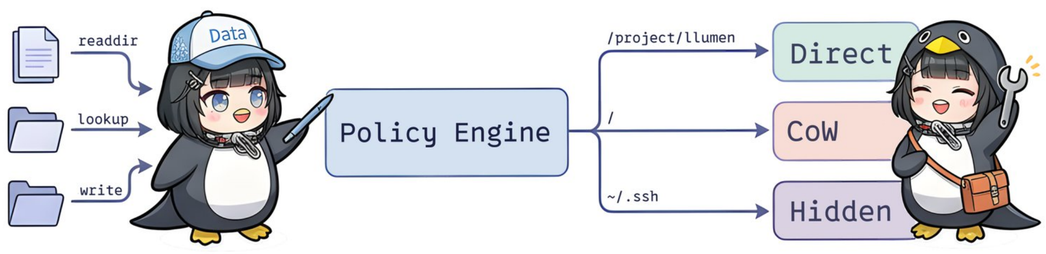
Write-ahead log
寫入快 3 倍

測試項目	原生	unix socket	WAL
ls_100	1.4ms	107ms	14.5ms
ls_500	2.7ms	622ms	_
ls_1000	4.5ms	1560ms	_
ls_5000	10ms	>10s	_
start up	_	85ms	13ms

lookup 在雙層資料夾的執行時間
_ 代表來不及做實驗

4 高細粒度檔案控制

每個 path 存取經由 policy.classify(path)
分派至三種 policy 之一



CoW 會在檔案寫入時複製，確保外部檔案不會被變更
並且維護 whiteout。
Hidden 適合機密的檔案，對於沙箱內部程式不可見。

5 結語

本專題開發出初步可用的沙箱，並提供靈活的檔案存取控制。
後續可從以下方向改進：
1. 加強檔案系統：優化 WAL(引入 fuzzy endpoint、改進 shm)、精簡 CoW 路徑，補上 splice 等。
2. 引入 userspace networking 與網路觀測。
3. 強化 syscall filter：讓更多程式能正確執行。
4. 提升 FUSE VFS 相容性：實作正確的 flock 機制。

逐步落實後，沙箱在效能與相容性上可以執行更多應用。