Bibliography

UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Progetto e realizzazione di una estensione VSCode per il debugging di un nucleo multiprogrammato

Relatore:

Prof: Giuseppe Lettieri

Prof: Luigi Leonardi

Candidato:

Francesco Mignone

ANNO ACCADEMICO 2023/2024

Abstract

Questo elaborato descrive la progettazione e l'implementazione di un'estensione per VS Code che facilita il debugging del nucleo multiprogrammato didattico. Gli obiettivi principali dell'estensione includono la possibilità di impostare e gestire breakpoints, visualizzare variabili in tempo reale e eseguire codice passo-passo. Per raggiungere questi obiettivi, l'estensione utilizza il Debug Adapter Protocol (DAP) per interfacciarsi con strumenti di debugging tradizionali come GDB.

Il lavoro presentato in questa tesi comprende un'analisi dettagliata dei requisiti funzionali e non funzionali, l'analisi dell'architettura dell'estensione e l'implementazione delle principali funzionalità. lo scopo dell'estensione è semplificare significativamente il processo di debugging del nucleo, offrendo un'interfaccia utente intuitiva e funzionalità avanzate di debugging.

Vengono inoltre discusse le limitazioni dell'estensione e vengono proposte possibili direzioni per futuri miglioramenti, tra cui l'aggiunta di ulteriori funzionalità e l'ottimizzazione delle prestazioni.

Indice

Capitolo 1

Introduzione

TODO: Forse emglio togliere la subsection

1.1 Contesto e motivazione

Il nucleo di un sistema operativo è il componente software fondamentale che gestisce le risorse hardware e software. A causa della sua complessità e della sua importanza critica, il debugging del nucleo è un compito altamente specialistico che richiede strumenti potenti e una profonda comprensione del sistema. Tuttavia, gli strumenti tradizionali come GDB e QEMU possono essere difficili da utilizzare, soprattutto per i nuovi sviluppatori o per coloro che preferiscono interfacce grafiche più intuitive.

Visual Studio Code (VSCode) è un editor di codice sorgente open-source sviluppato da Microsoft. VS Code è diventato molto popolare tra gli sviluppatori grazie alla sua facilità d'uso e alla sua capacità di supportare molti linguaggi di programmazione grazie alla sua vasta gamma di estensioni disponibili. Tuttavia, al momento della scrittura di questa tesi, non esiste un'estensione dedicata per il debugging del nucleo didattico su VSCode.

L'obiettivo di questa tesi è sviluppare un'estensione per VS Code che renda il debugging del nucleo più accessibile e intuitivo, permettendo agli sviluppatori di beneficiare dell'ambiente user-friendly di VSCode senza rinunciare alla potenza degli strumenti tradizionali.

Capitolo 2

Ambiente e strumenti

2.1 Il debugger - GDB e QEMU

2.1.1 Quick emulator - QEMU

QEMU é un emulatore open-source, permette di emulare l'architettura di un processore. Permette quindi l'utilizzo di vari sistemi operativi ad un livello di virtualizzazzione Kernel-Based (Kernel Vitual Machine - KVM) con il beneficio di prestazioni vicine all'hardware. Per il nostro utilizzo QEMU emula un sistema x86-64.

2.1.2 GNU Debugger - GDB

GNU Debugger (GDB) è un debugger portatile, permette quindi di testare e eseguire il debug di programmi. Eseguire il programma in questo ambiente controllato permette al programmatore di tenere traccia dell'esecuzione e monitorare le risorse al fine di individuare un eventuale malfunzionamento nel codice. Per la realizzaione dell'estensione utilizzeremo la funzione di debug remoto per connetterci ad un socket di sistema utilizzato da QEMU per il debug. GDB utilizza delle chiamate di sistema chiamate process trace (ptrace).

Breakpoints

Un breakpoint permette al programma in esecuzione all'interno di un debugger di interrompere il flusso in un determinato punto. Si realizzano sostituendo all'istruzione, alla quale si vuole fermare l'esecuzione, una speciale istruzione la quale solitamente invia un segnale SIGTRAP, il quale verrà catturato dal debugger. Il procedimento di sostituzione è eseguito dal debugger stesso prima di avviare l'esecuzione, nel caso di GDB il programmatore deve eseguire il comando break [arg] dove l'argomento può essere la specifica linea di codice o un simbolo. All'interno di VSCode vi é la possibilità di inserire breakpoint

cliccando sul lato sinistro della riga di codice interessata, si occuperá poi di comunicare al Debug Adapter (TODO: inserisci hyperlink a chapter del DP) la richiesta di inserimento dell'interruzione.

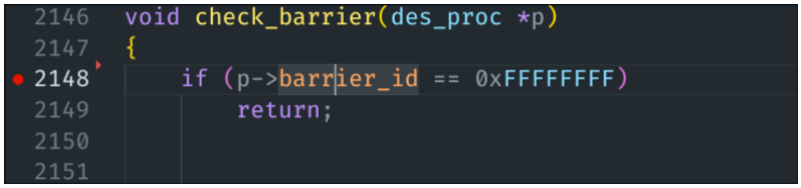

Figura 2.1: Esempio di un breakpoint in VSCode

Continue Stop

Step Over

Step In

Step Out

Watch delle variabili

Call Stack

2.2 L'architettura del debugger di VS Code

2.2.1 Debug Adapter - DP

2.2.2 Debug Adapter Protocol- DAP

2.2.3 Estensione CppTools

utilizzeremo quindi un DP già realizzato dall'estensione di CppTools

2.3 Webview di VS Code

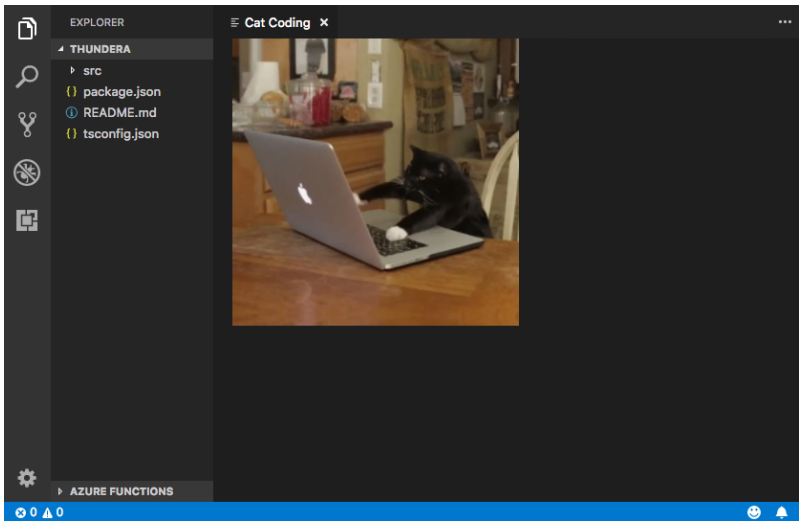

Figura 2.2: Esempio di una webview in VSCode

Capitolo 3

Requisiti dell'estensione di debug

3.1 Funzionalità di debug

3.2 Analisi dello stato del nucleo

Capitolo 4

Implementazione

4.1 Event listener

ci scrivi del onDebugSessionStart

4.2 Webview

ci scrivi della classe e i suoi metodi

4.2.1 Update Webview

4.2.2 Custom Debug Adapter request

Capitolo 5

Utilizzo del debugger

5.1 Breakpoint

5.2 Continue Pause

5.3 Step Over, Step In e Step Out

5.4 Informazioni aggiuntive

Capitolo 6

Conclusioni

Capitolo 7

Sviluppi futuri

7.1 Codice

Appendice A

Ringraziamenti