Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

TITOLO ITALIANO

TITOLO INGLESE

NOME CANDIDATO

Relatore: *Relatore*
Correlatore: *Correlatore*

Anno Accademico 2014-2015

Nome Candidato: *Titolo italiano,* Corso di Laurea in Informatica, © Anno Accademico 2014-2015

INDICE

ELENCO DELLE FIGURE

"Inserire citazione"
— Inserire autore citazione

1

INTRODUZIONE A SEL4A

Un sistema operativo (SO) è un insieme di software che gestisce le risorse hardware e software di un sistema di elaborazione fornendo servizi agli applicativi utente.
In un computer quindi il sistema operativo fornisce l'unica interfaccia diretta con l'hardware e in quanto tale ha un accesso esclusivo con il massimo dei privilegi detto *kernel mode*. Questo comporta che una vulnerabilità all'interno del sistema operativo può portare a gravi conseguenze per l'integrità e la sicurezza del sistema, inoltre qualche malintenzionato potrebbe approfittare di questo bug per trarne profitto. Uno degli obiettivi principali di un SO è quindi quello di garantire la sicurezza, ulteriore scopo è l'efficienza: un buon sistema operativo deve saper sfruttare al meglio tutte le risorse che ha a disposizione, dalla gestione della memoria per sfruttare al meglio lo spazio alla schedulazione dei processi per ottimizzare i tempi di esecuzione. Come ultimo obiettivo, ma non per questo meno rilevante, deve rendere il più semplice possibile l'utilizzo del dispositivo su cui è installato. Dalla definizione di SO data a inizio capitolo possiamo isolare una specifica parte di codice che è quella che permette al software di interfacciarsi con l'hardware, quindi l'accesso e la gestione delle risorse di un dispositivo, questa specifica parte si chiama *kernel* che come suggerisci il nome (nocciolo dall'inglese) rappresenta la parte centrale di un sistema operativo su cui tutto il resto si appoggia.

1.1 MICROKERNEL E KERNEL MONOLITICI

Esistono vari modelli strutturali per i sistemi operativi: monolitici, modulari, a livelli, microkernel ed ibridi, ad oggi i più diffusi sono gli ibridi, che combinano i vari modelli tra di loro, ma che in gran parte si basano su sistemi monolitici i quali consistono di un unico file binario statico al cui interno sono definite tutte le funzionalità del kernel e che viene eseguito in un unico spazio di indirizzi, questo comporta dei vantaggi:

- efficienza → motivo principale per cui la maggior parte dei sistemi operativi ancora si basano su kernel in gran parte monolitici, lavorando nello stesso spazio di indirizzi e gestendo tutto attraverso chiamate a procedura il SO risulterà molto reattivo e performante

- semplicità → in quanto non ha una vera e propria strutturazione ma il codice è tutto in un unico file binario risulta chiaramente più semplice da progettare anche se poi l'implementazione risulta difficile

d'altra parte ha anche degli svantaggi:

- inserimento di un nuovo servizio → questo richiede la ricompilazione del kernel, quindi non permette l'inserimento di un nuovo servizio a runtime (problema risolto nei modelli ibridi)

- dimensione → dovendo gestire tutte le principali funzionalità del sistema operativo il kernel sarà composto da milioni di righe di codice sorgente (MSLOC - linux ha circa 20MSLOC) e questo porta direttamente al successivo grosso svantaggio

- sicurezza → maggiore è il numero di righe di codice maggiore sarà il numero di possibili bug, essendo tutto il codice eseguito nello stesso spazio di indirizzi un bug rischia di far bloccare l'intero sistema anche se il problema è molto piccolo e isolato a una minima funzione del kernel.

All'estremo opposto troviamo i *microkernel* che sono composti da un kernel ridotto al minimo indispensabile, che comprende la gestione della memoria, dei processi e della CPU, le comunicazioni tra processi (IPC) e l'hardware di basso livello, mentre tutto il resto deve essere gestito da server (deamon) che operano sopra al kernel, quindi in spazi di indirizzi separati.
I microkernel sono spesso usati in sistemi embedded in applicazioni

mission critical di automazione robotica o di medicina, a causa del fatto che i componenti del sistema risiedono in aree di memoria separate, private e protette.
Anche questo modello ha dei vantaggi:

- flessibilità → l'inserimento di un nuovo servizio avviene al di sopra del kernel quindi in qualsiasi momento è possibili aggiungere o togliere servizi senza dover modificare il kernel.

- sicurezza → minore quantità di codice eseguita in kernel mode (quindi minore quantità di bug e minore superficie attaccabile) maggiore è la sicurezza del sistema, inoltre i servizi sono lavorano in uno spazio di indirizzi differente da quello del kernel di conseguenza se un server (su cui viene eseguito un servizio) smette di funzionare tutto il resto del sistema continua a funzionare normalmente e si potrà procedere a riavviare quel singolo servizio

- semplicità → essendo il codice composto da giusto qualche decina di migliaia di righe di codice (KSLOC) risulta molto più facile da scrivere

e dall'altro lato ha un grande svantaggio:

- efficienza → dato che ogni servizio gira a livello utente l'utilizzo di uno qualsiasi di questi richiede il ricorso a chiamate di sistema che rallentano fortemente l'esecuzione di ogni operazione, motivo principale per cui ancora oggi i sistemi operativi si basano in gran parte su sistemi monolitici.

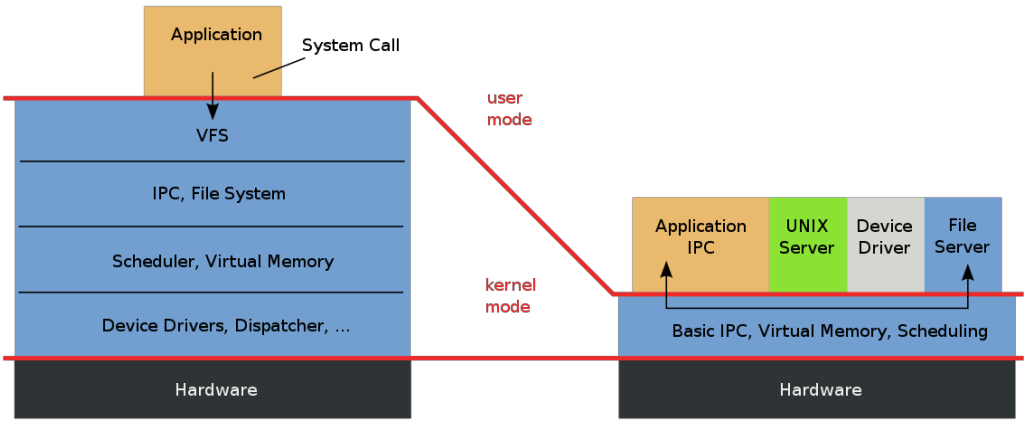

Figura 1: Kernel monolitici (sinistra) vs microkernel (destra)

1.2 SEL4

seL4 fa parte della famiglia dei microkernel L4 che risalgono alla prima metà degli anni '90 creato da Jochen Liedtke e ad oggi fa parte del Trustworthy System per sopperire alle scarse performance dei primi sistemi operativi basati su microkernel.
seL4 è un microkernel non un sistema operativo, infatti non fornisce nessun servizio che siamo solitamente abitati a trovare su un comune SO ma "è solo un sottile involucro attorno all'hardware" [1], tutti i servizi devono essere eseguiti in modalità utente, chiaramente questi dovranno essere importati ad esempio da sistemi operativi open-source come Linux (oppure scritto da zero).

BIBLIOGRAFIA

[1] Gernot Heiser. The sel4 microkernel - an introduction. *The seL4 Foundation*, Revision 1.2, 2020. (Cited on page 10.)

[2] Wikipedia, l'enciclopedia libera. Kernel, 2023. ultima modifica 7 giu 2023.

[3] Wikipedia, l'enciclopedia libera. Operating system, 2023. ultima modifica 16 luglio 2023.

[4] Rosario Pugliese. Chiedere al prof come citare le sue slide.