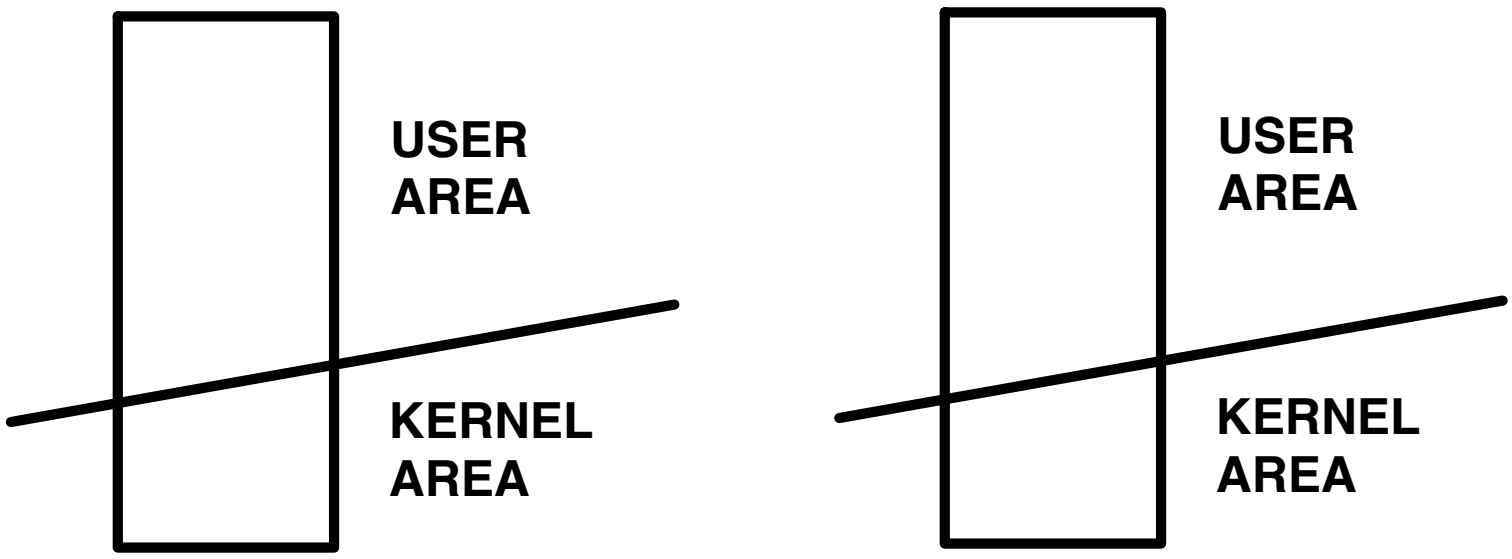
Condivisione delle pagine kernel

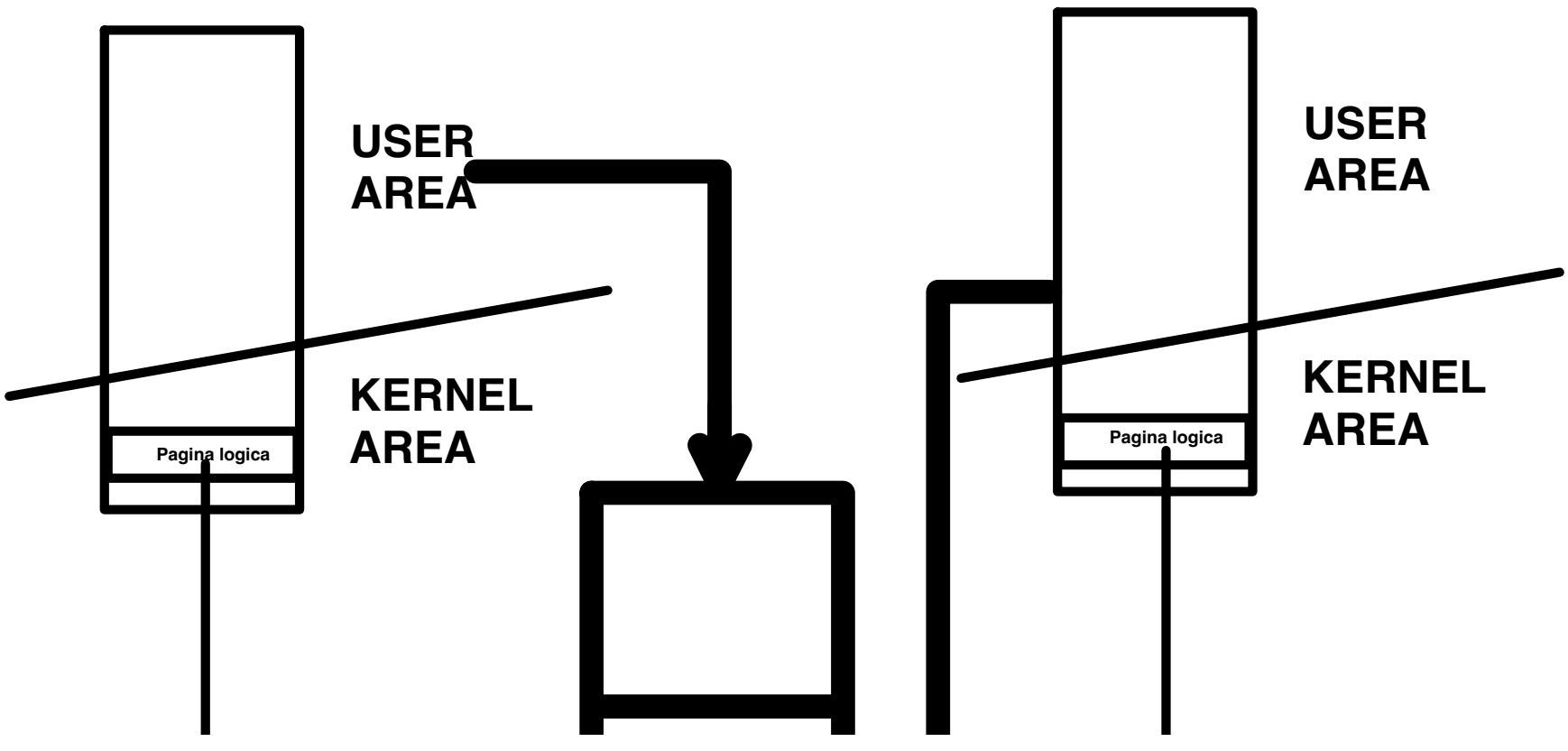
La paginazione ci fornisce ulteriori vantaggi nel momento in cui noi lavoriamo su un sistema operativo moderno. Se noi guardiamo un address space di un'applicazione noi sappiamo che esso è l'insieme degli indirizzi logici che possono essere utilizzati quando l'applicazione è in esercizio.
Supponiamo che questa applicazione ha un thread in esercizio che ad un certo punto chiama il kernel: quando questo thread comincia ad eseguire un blocco di codice del kernel, quest'ultimo utilizza indirizzi logici, quindi quando andiamo ad eseguire in modalità kernel, anche qui accediamo a delle PAGINE LOGICHE, non a delle pagine fisiche, il codice esprime degli accessi a pagine logiche, dove ovviamente anche l'IR del mio processore sta esprimendo accessi a pagine logiche.
Questo implica dire che queste pagine logiche devono far parte del contenitore che è accessibile a questa applicazione.
Infatti questo contenitore tipicamente all'interno di un sistema operativo è diviso in due zone: una zona USER e una zona KERNEL.

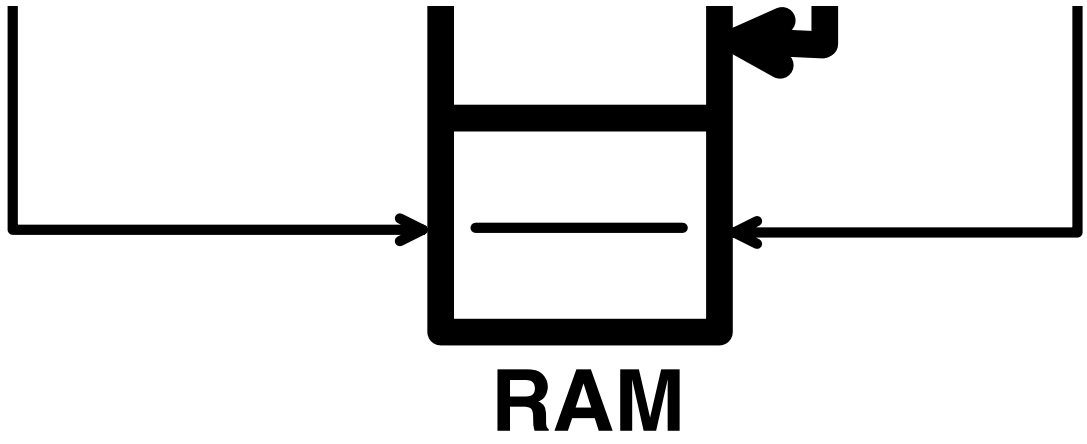


Address Space

Nella zona kernel ci sono pagine accessibili in modalità Kernel, e nella zona user pagine accessibili in modalità utente.

Nella Kernel Area ci sono le pagine logiche di livello KERNEL e chiaramente per sapere, quando lavoriamo in modo kernel, dove queste pagine (logiche) sono in memoria fisica, passiamo comunque attraverso la PAGE TABLE, ma non solo.
Supponiamo di avere due AB di un'applicazione attiva fatto con questa suddivisione.
In memori fisica, questo kernel, quante volte è presente? Mentre le informazioni USER potrebbero essere presenti in maniera disparata in memoria fisica, quindi la parte user della prima applicazione è presente in una zona e la parte user della seconda in un'altra (ovviamente adesso prescindiamo dal fatto che queste pagine USER possono essere non contigue), ma il kernel è presente una sola volta.
In realtà il kernel è uno solo su una istanza di sistema che noi abbiamo attiva. Quando il kernel prende il controllo e vuole toccare una struttura dati che è in una certa pagina logica, la stessa informazione in memoria fisica viene toccata se vogliamo accedere alla stessa informazione anche dall'applicazione B.

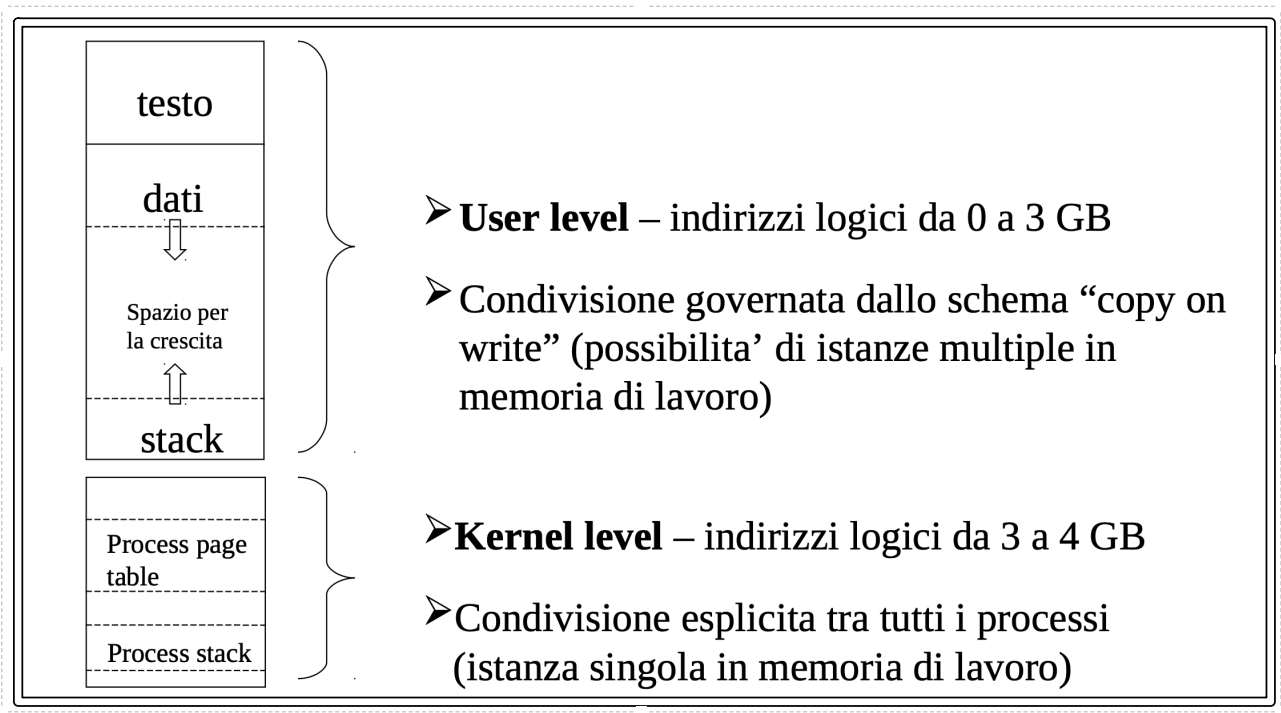




Quindi il kernel è uno solo all'interno della memoria fisica, utilizza alcuni frame, e ovviamente tramite la tabella delle pagine permettiamo la condivisione della kernel area. Tutto ciò che è la rappresentazione delle pagine logiche del kernel in memoria fisica.
Ovviamente le tabelle delle pagine hanno anche informazioni che vanno ad indicare se una pagine è una pagina user o una pagina kernel, in modo tale che se il processore sta lavorando in modalità user, quindi stiamo gestendo un'istruzione a livello user nell'address space e questa istruzione esprime la volontà di accedere ad una pagina del kernel, quindi alle strutture dati del kernel, la page table non permette questo passaggio e quindi abbiamo una trap che ci riporta il controllo al sistema operativo.

- la condivisione tramite segmentazione e paginazione è anche un modo per rendere accessibile a processi distinti l'unica istanza di codice/dati del kernel
- il kernel è infatti tipicamente raggiungibile in un determinato set di indirizzi logici
- questi vengono "mappati" su indirizzi fisici identici per tutti i processi attivi in modo da ottenere una visione univoca e coerente dello stato del kernel stesso
- le tabelle delle pagine contengono anche informazioni (bit di controllo) indicanti la possibilità di accedere a date pagine in modalità user oppure kernel

In un address space abbiamo la condivisione della ZONA KERNEL (di pagine del kernel) nel momento in cui lavoriamo all'interno di questo address space.



Qui c'è la rappresentazione di un address space su processori X86 per sistema operativo linux nel momento in cui lavoriamo a 32 bit.
Con 32 bit abbiamo il limite di 4GB che possiamo esprimere all'interno di un address space, quindi questo AB globalmente è fatto da 4 GB.
La struttura di questo address space per come la gestisce linux è composta da 0 a 3 GB (in termini di offset) per la zona USER, e sotto abbiamo la zona KERNEL. Nel kernel abbiamo la page table per questo processo, ma anche di altri, infondo è solo un'istanza singola nella memoria di lavoro.
Nel kernel manteniamo tutto ciò che serve per gestire qualsiasi processo. Se abbiamo un thread attivo, in questa zona del kernel avremo anche il suo stack e abbiamo tutti gli stack di tutti i thread attivi.
Sulla parte USER possiamo comunque continuare a condividere informazioni in memoria fisica, con lo schema "Copy On Write", per cui se questa applicazione è tale per cui che ad un certo punto nel testo andiamo a chiamare un servizio del kernel (fork), questo ci permetterà di generare un nuovo processo in cui noi condividiamo le nostre pagine utilizzando "Copy on write": il primo che va a scrivere da qualche parte. Ci sarà una trap e riprende il controllo il software del kernel, riserva la pagina e fa una copia privata di quella pagina logica all'interno di quella pagina fisica riservata, e aggiorna ovviamente le page table.