De-mappatura/gestione-protezione di pagine in sistemi UNIX

martedì 16 maggio 2023 17:36

De-mappatura/gestione-protezione di pagine in sistemi UNIX

int munmap(void *addr, size_t length)

Descrizione de-mappa una regione di pagine contigue

Argomenti 1) *addr: inizio della regione da un-mappare
2) lenght: taglia della regione da un-mappare

Restituzione -1 in caso di fallimento

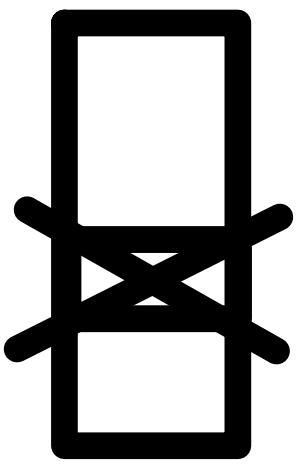
int mprotect(void *addr, size_t length, int prot)

Descrizione gestisce il livello di protezione di pagine

Argomenti 1) *addr: inizio della regione da gestire
2) lenght: taglia della regione da gestire
3) prot: tipo di protezione da applicare (PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE)

Restituzione -1 in caso di fallimento

La prima, la MUNMAP, accetta come primo parametro un indirizzo di memoria all'interno del contenitore, tipicamente l'indirizzo dell'inizio di una pagina o più pagine, e come secondo parametro specifico una taglia, una quantità di byte.
A partire da quell'indirizzo chiedo un UNMAP della memoria.
Dato un address space, ad un certo punto una pagina che sta sotto che avevo mappato precedentemente ed anche utilizzando, non mi serve più.



Address Space

Non soltanto non mi serve più la corrispettiva informazione in memoria fisica, ma nemmeno l'informazione all'interno dello spazio di indirizzamento logico.

Quindi passo l'indirizzo di questa pagina, passo una quantità di byte che voglio unmappare e così quella pagina viene rimossa dall'address space.
Da questo oggetto logico viene rimosso il corrispettivo contenuto fisico in termini di informazione raggiungibile.
E la page table di quel processo viene aggiornata per sganciarci dall'utilizzo del corrispettivo frame fisico associato alla ex pagina logica.

Abbiamo anche la possibilità di cambiare la protezione delle zone di memoria passando ancora un indirizzo come primo parametro, una taglia come secondo parametro e come terzo parametro una "regola di protezione" che deve essere applicata ORA.
Possiamo definire READ_ONLY per una certa zona, WRITE_ONLY, EXEC o NONE.
Chiaramente possiamo anche combinare queste informazioni tra di loro per avere, per esempio, che una zona che originariamente era solo leggibile, diventi READ/WRITE, e quindi in questo ultimo parametro passiamo l'OR logico (|) tra READ e WRITE.

Il fallimento è dovuto al fatto che stiamo cercando di modificare la protezione di una zona (di cui passiamo l'indirizzo come primo parametro) che in realtà non è mappata.