domande esame: Sistemi Operativi

Sistemi Operativi

Alma Mater Studiorum – Università di Bologna (UNIBO)

15 pag.

TEORIA SISTEMI OPERATIVI

YYY01

1. In linguaggio ANSI C, sia dato il seguente programma:

```
#include <stdlib.h>
int main (void) {
     char str[120] = “ciao”;
     char *p;
     p = &str;
     return (0);
}
```

Il programma viene correttamente compilato e linkato. Procediamo ad eseguirlo. Durante l’esecuzione, alla variabile p viene assegnato l’indirizzo della stringa str.

Dopo l’assegnamento, l’indirizzo contenuto nella variabile p specifica un indizio in memoria fisica oppure in memoria virtuale?

Specifica un indirizzo di memoria virtuale che in fase di esecuzione verranno mappati nei corrispondenti indirizzi di memoria fisica dal Memory Management Unit.

2. Definire cos’è un Sistema di Elaborazione dell’Informazione.

Un Sistema di Elaborazione dell’Informazione è una macchina (digitale, automatica, elettronica, programmabile) in grado di elaborare dei dati.

Quali sono i componenti HW di un computer?

CPU esegue istruzioni, effettua calcoli, controlla input/output, coordina spostamenti di dati.

BUS trasferisce i dati tra la memoria e i dispositivi.

Memoria RAM mantiene programmi quando devono essere eseguiti.

Memoria ROM contiene il BIOS

Dischi mantengono enormi quantità di dati e programmi anche dopo lo spegnimento

Periferiche I/O

3. Che cosa accade durante in bootstrap? E dove si trovano le istruzioni che guidano il bootstrap?

L’avvio del Bootstrap si divide in due fasi:

1)La macchina si accende eseguendo il programma di bootstrap che è già in memoria. Il sistema operativo è memorizzato nella memoria di massa.

2)Il programma di bootstrap dirige il trasferimento del s.o. nella memoria principale e poi le trasferisce il controllo.

4. Una stringa è collocata in memoria in modo diverso se la CPU è di tipo Big Endian invece che Little Endian?

Sì, In Little Endian i Byte più significativi del numero sono collocati dei Byte di indirizzo maggiore.

In Big Endian I Byte più significativi del numero sono collocati nei Byte di indirizzo minore.

5. Dichiarare una struttura dai pacchettizzata.

Una struttura dati di questo tipo è un struttura che evita lo spreco in memoria di spazio occupando solo lo spazio necessario per contenere i singoli membri della struttura e non a multipli di WORD.

```
struct struttura {
     uint32_t i;
     uint8_t c;
} _attribute_((packed))STRUTTURA;
```

6. Sapendo che una CPU lavora “a 64 bit” possiamo dire qualcosa a proposito di qualche BUS del computer?

Una Word è un blocco di byte in memoria avente la stessa di dimensione dell’ampiezza di byte del Bus dei Dati. Questo indica quanti byte possono essere trasferiti, al massimo, con una unica operazione del bus tra CPU e memorie.

7. Cos’è e a cosa serve il DMA?

Il DMA è un meccanismo hardware dedicato al trasferimento dati che consente di ridurre al minimo il coinvolgimento della CPU.

Il trasferimento dei dati, dal disco alla memoria, effettuato dal DMA non coinvolge la CPU tranne che:

- Quando la CPU ordina al DMA lo spostamento del blocco dati.

- Quando il DMA avvisa la CPU dell’avvenuto trasferimento.

8. A cosa serve l’opzione -I del gcc?

L’opzione -I serve per indicare la posizione dei file header.

9. A cosa serve l’opzione -L del gcc?

L’opzione -L serve per indicare la posizione delle librerie (statiche o dinamiche).

A cosa serve l’opzione -Wl,-rpath,RunTimeLibDir del gcc?

L’opzione -Wl,-rpath,RunTimeLibDir serve per indicare la posizione delle librerie dinamiche a run-time.

Cos’è il modo Kernel?

Il modo kernel è una modalità di esecuzione che permette di usare funzionalità pericolose del processore. È spezzato un più parti perché ha diversi livelli di privilegio.

10. Cosa sono i Livelli di Privilegio (Rings) della CPU?

Indicano a quale livello del Sistema Operativo ogni processo può accedere in fase di lettura o scrittura e quali operazioni può effettuare.

11. Un Task può avere più stack a sua disposizione? Perché si o perché no?

Sì, perchè è presente uno stack per le funzioni utente nello spazio utente e più stack per l’esecuzione delle system calls nello spazio kernel.

12. Cos’è una System Call?

Le system call sono servizi messi a disposizione in diverse modalità. Le system call sono disponibili mediante chiamate ad interrupt effettuabili in assembly.

Quali sono i registri specializzati?

Registri Specializzati (nome differente a seconda del processore):

- *Extended Instruction Pointer (EIP) serve a formare l'indirizzo in memoria (detto Program Counter (PC)) della prossima istruzione da eseguire. Contiene l'offset della prossima istruzione da eseguire, rispetto all'inizio del segmento di codice.*
- *Program Status (PS) detto anche Registro di Stato (Status Register).*
- *Stack Segment Register (SS). Punta all'inizio dello stack.*
- *Code Segment Register (CS). Punta all'inizio della sezione text, il codice eseguibile.*
- *Data Segment Register (DS). Punta all'inizio della sezione data, con le var globali.*

Descrivere il supporto ai modi di esecuzione della CPU per i processi.

Quando un processo esegue una chiamata ad interrupt, la CPU entra nella modalità di esecuzione kernel e deve utilizzare uno (o più) stack "di sistema" separato rispetto allo stack utilizzato per le normali chiamate a funzione.

Ciò permette al kernel di separare e proteggere i record di attivazione utilizzati nelle chiamate ad interrupt.

Ciascun processo, perciò, mantiene uno stack per ciascun livello di privilegio della CPU.

Che cosa sono i Task?

I processi, gli interrupt e le eccezioni eseguono ciascuno in un proprio contesto denominato Task. I Task mantengono le informazioni sui segmenti di memoria utilizzati e sul valore corrente dei registri principali.

13. Qual'è il compito del loader?

Quando un programma viene lanciato, il loader (che fa parte del sistema operativo) :

- *crea i segmenti di memoria per il processo*
- *carica in memoria il codice eseguibile del programma*
- *carica in memoria la sezione Data*
- *crea lo stack utente per l'esecuzione delle funzioni del programma*
- *crea gli stack di sistema per l'esecuzione delle system calls che verranno invocate dal programma utente*
- *copia eventuali argomenti passati a riga di comando mettendoli a disposizione del main*
- *infine ordina alla CPU di eseguire la prima istruzione eseguibile corrispondente al main del programma*

14. Indicare la differenza tra FAULT e ABORT e dare un esempio per ciascuna delle due eccezioni.

Entrambe sono delle eccezioni.

- *FAULT: l'istruzione che ha causa il trap viene interrotta e viene eseguita la routine di gestione. In un successivo momento l'istruzione verrà riseguita da principio. (Esempio Page fault, la pagina di memoria che contiene l'indirizzo a cui stiamo cercando di accedere è*

swappata su disco. La pagina viene caricata in memoria. Prima o poi l'istruzione verrà nuovamente eseguita.)

- *ABORT: Causata da errore irrimediabile, l'istruzione viene abortita e il processo killato (divisone di un numero per zero, segmentation fault).*

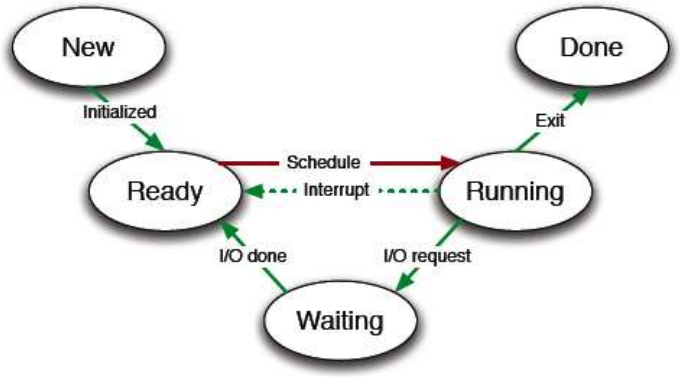
15. Spiegare la differenza tra Interrupt Sincroni e Interrupt Asincroni

Gli interrupt sincroni sono interrupt software esplicitamente chiamati dalla CPU tramite l'istruzione INT n. Quelli asincroni, sono hardware, e sono scatenati da una periferica mediante il BUS che avvisa la CPU che è accaduto un evento da gestire.

Spiegare la differenza tra Interrupt Mascherabili ed Non Mascherabili

- *I Mascherabili possono essere rimandati*
- *Non Mascherabili non possono essere rimandati.*

16. Disegnare il diagramma di stato dei processi, dal punto di vista dello scheduler.



17. Spiegare in che situazione un processo si trova in stato di waiting.

Un processo si trova in stato di waiting il processo effettua una chiamata a sistema bloccante (richiesta I/O).

18. Nella bash, qual'è la differenza tra variabili locali e variabili d'ambiente?

Una variabile locale è una variabile visibile e modificabile solo all'interno della shell che l'ha creata. Mentre una variabile d'ambiente di una shell viene eredita mediante copia dalle sue subshell. Se una subshell modifica tale variabile ereditata comunque non viene modificata solo la copia ereditata.

19. Nella bash, qual'è la differenza tra un variabile vuota e una variabile che non esiste? È possibile eliminare una variabile?

Una variabile vuota è una variabile che è stata creata ma non inizializzata mentre una variabile che non esiste non è ancora stata creata. È possibile eliminarla con il comando unset nomevariabile.

20. Cos'è la funzione di Autocompletamento della bash?

È una funzionalità che mette a disposizione bash per facilitare l'inserimento dei comandi. Per utilizzarla basta scrivere le prime lettere del comando in questione e poi premere TAB.

Descrivere il contesto di un processo.

Ogni processo ha il proprio contesto, ovvero il proprio

- *Process ID,*
- *Program Counter,*
- *Stato dei Registri,*
- *Stack,*
- *Codice,*
- *Dati,*
- *File Descriptors,*
- *Entità IPC,*
- *Azioni dei segnali.*

21. Che cosa hanno in comune i PosiX thread di uno stesso processo?

I PosiX thread di uno stesso processo hanno in comune le variabili globali.

22. Se un processo possiede diversi PosiX thread e quel processo genere un processo figlio, il processo figlio possiede dei thread?

Si, il processo figlio eredita una copia identica dell'ambiente di lavoro del processo padre a partire dal momento della creazione. Il processo figlio eredita un solo thread dal genitore, cioè il thread che aveva chiamato la fork().

23. Se un processo possiede diversi posix thread e quel processo genere un processo figlio, i thread del processo figlio "vedono" le variabile globali del processo padre oppure no?

In generale si può dire che le variabili globali non sono condivise fra processo padre e processo figlio, tuttavia il sistema operativo in realtà condivide inizialmente le pagine, perché fork() implementa un meccanismo copy-on-write, il che significa che a condizione che nessuno dei processi modifichi le pagine, esse sono condivise.

24. Che cos'è errno?

errno è una variabile di errore che ogni thread può avere. Contiene un valore significativo solo se la chiamata è stata effettuata con un errore. Per utilizzare questa variabile dobbiamo far uso della libreria errno.h.

Descrivere i Vantaggi e gli Svantaggi dei Thread.

Vantaggi:

- *Visibilità dei dati globali: condivisione di oggetti semplificata.*
- *Più flussi di esecuzione*
- *gestione semplice di eventi asincroni (I/O per esempio)*
- *Comunicazioni veloci. Tutti i thread di un processo condividono lo stesso spazio di indirizzamento, quindi le comunicazioni tra thread sono più semplici delle comunicazioni tra processi.*
- *Context switch veloce. Nel passaggio da un thread ad un altro di uno stesso processo viene mantenuto buona parte dell'ambiente.*

Svantaggi:

- *Concorrenza invece di parallelismo. Occorre gestire la mutua esclusione per evitare che più thread utilizzino in maniera scoordinata i dati condivisi, modificandoli in momenti sbagliati.*

25. Se un primo thread esegue una chiamata a funzione di libreria del C, e questa produce un errore e scrive il codice di errore nella variabile errno, un altro thread può controllare il contenuto della variabile errno per conoscere il tipo di errore capitato nel primo thread?

No, la variabile errno è specifica per ogni thread

26. Definire la Liveness.

Per Liveness si intende la capacità di un programma di concorrente di far "vivere" e progredire nel tempo i propri thread fino al compimento delle operazioni relative, evitando condizioni che possano determinarne un blocco (ad esempio Deadlock e Starvation).

27. Definire il Busy Waiting.

Il Busy Waiting è un meccanismo di sincronizzazione nel quale un thread che attende il verificarsi di una condizione, lo fa verificando continuamente se la condizione sia diventata vera.

28. Definire il Deadlock.

Stallo o blocco. Due o più thread aspettano indefinitamente un evento che può essere causato solo da uno dei thread bloccati.

29. Definire la Starvation.

Con Starvation si intende l'attesa indefinita da parte di un processo di accedere ad una

risorsa perché essa viene continuamente detenuta da processi con priorità maggiore.

30. Che cosa sono i processi Zombie?

Un processo zombie è un processo informatico che, nonostante abbia terminato la propria esecuzione, possiede ancora un PID ed un Process control block, necessario per permettere al proprio processo padre di leggerne il valore di uscita.. Fintanto che il processo padre non esegue la wait, il processo figlio viene definito zombie.

31. Descrivere concisamente qualche tipo di strumento messo a disposizione dallo standard

POSIX per sincronizzare tra loro i processi.

Le IPC a memoria condivisa (shared memory) forniscono ai processi cooperanti una porzione di memoria in comune in cui possono scrivere e leggere in modo da passarsi dei dati.

Le IPC a scambio di messaggi (message passing) forniscono ai processi la possibilità di inviare e ricevere messaggi con cui scambiarsi i dati di interesse.

Cos'è il PCB?

Un sistema operativo ha strutture dati apposite con cui tiene traccia di tutti i processi in esecuzione e del loro stato. In particolare per ogni processo viene creato un Process Control Block (PCB) o task control block ove si mantengono informazioni specifiche sul processo e il suo stato.

32. Spiegare cos'è la rilocazione statica.

È l'operazione svolta dal linker che modifica le istruzioni di salto cambiando l'operando che specifica l'indirizzo a cui saltare per specificare correttamente l'indirizzo in cui si trova l'istruzione da eseguire ora che i diversi moduli sono stati accorpati. La rilocazione statica riguarda tutti i salti assoluti.

33. Spiegare che cos'è il binding degli indirizzi.

Si definisce binding degli indirizzi la sequenza con cui ad un indirizzo logico, specificato dalla coppia <SEGMENTO:OFFESET>, viene associato l'indirizzo fisico in memoria RAM fisica. Nei sistemi moderni viene effettuato utilizzando i meccanismi di segmentazione, paginazione e memoria virtuale.

TEORIA SISTEMI OPERATIVI

YYY02

1. In linguaggio ANSI C, siano dichiarate le seguenti variabili, e siano queste variabili debitamente inizializzate:

```
pthread_mutex_t mutex;
pthread_cond_var cond;
```

considerare poi la seguente porzione di codice, in cui condizione sia una espressione da valutare:

```
while (condizione) {
     pthread_cond_wait (&cond, &mutex);
}
fail_qualcosa();
```

La porzione di codice, sopra riportata, effettua busy-waiting?

No, perché wait interrompe l'esecuzione rilasciando la mutua esclusione e mettendosi in attesa del segnale specificato da cond.

2. Una stessa struttura dati ha la stessa dimensione in tutti i processori? Perché?

No, perché dipende da processore e molto spesso i registri general purpose dei processori hanno la stessa dimensione del Bus dei Dati.

3. In un programma in C scrivo una funzione in cui dichiaro un vettore di elementi, i cui elementi sono un tipo di struttura dati struct A. Guardando come viene collocato in memoria quel vettore, può accadere che tra un elemento ed il successivo, dello stesso vettore, ci siano dei byte non occupati dalla struttura dati di tipo struct A? Perché?

Si, se la dimensione della struttura non è un multiplo di una word.

4. In linguaggio C, se un programma è costituito da più moduli, è possibile che una stessa funzione (cioè una funzione con stesso nome, stessi tipi di argomenti e stesso tipo di dato restituito) sia implementata in due diversi moduli senza che questo provochi un errore nella generazione del programma eseguibile? Perché?

Sì, è sufficiente che le funzioni che sono implementate in moduli differenti siano dichiarate "static" così da ridurne lo scope al solo modulo a cui appartengono.

5. La seguente istruzione si trova nella funzione main di un modulo scritto in C.

```
ris = PRODOTTO (3-4, SOMMA(2,7));
```

Precedentemente, nello stesso file, le due macro sono state così definite:

```
#define PRODOTTO (X,Y) ((X)*(Y))
#define SOMMA (X,Y) ((X)+(Y))
```

Se quel modulo viene processato dal preprocessore del compilatore gcc, in che modo viene tradotto quella istruzione dal preprocessore?

ris = ((3-4)((2)+(7)))*

6. Supponiamo di inserire le seguente due istruzioni, in un modulo scritto in C, in una zona fuori da tutte le funzioni del modulo;

```
#define NUM 10
int vet[10];
```

Se quel modulo viene processato dal preprocessore del compilatore gcc, in che modo viene tradotta quella istruzione dal preprocessore?

Il preprocessore sostituisce tipograficamente, ogni volta che compare, 10 al posto di NUM, ma in questo caso non fa nulla perché non compare mai.

7. Spiegare cosa si intende con Race Condition.

Indica la situazione in cui più thread o processi concorrenti accedono (e manipolano) dati condivisi senza controlli di sincronizzazione. In una race condition, il contenuto finale dei dati condivisi dipende dalla sequenza di esecuzione e può causare una inconsistenza dei dati dal punto di vista del singolo thread.

8. Spiegare cosa si intende quando si dece che una sequenza di istruzioni viene eseguita in maniera "atomica".

Una operazione si dice atomica se è indivisibile, ovvero se nessun'altra operazione può cominciare prima che la prima sia finita, è quindi non può esserci interleaving. Il risultato di quella operazione è sempre lo stesso se parte dalle stesse condizioni iniziali.

9. Quando un processo esegue una fork, le sue pagine in memoria vengono immediatamente duplicate?

fork() implementa un meccanismo copy-on-write, il che significa che a condizione che nessuno dei processi modifichi le pagine, esse sono condivise. Un processo o task può avere più thread e ad ognuno di essi è associato uno stack.

10. Spiegare come dovrebbe funzionare l' "algoritmo ottimale" di sostituzione delle pagine.

L'algoritmo ottimale è quello che rendere minimo il numero di page fault che avvengono complessivamente. L'algoritmo funziona scaricando la pagina che non verrà utilizzata per il periodo di tempo più lungo. È però impossibile da implementare perché richiede conoscenza dei riferimenti richiesti nel futuro. Viene utilizzato come lower bound.

11. Spiegare come funziona l'algoritmo "Least Recently Used (LRU)" di sostituzione delle pagine.

L'algoritmo opera in modo da sostituire la pagina che è stata acceduta meno recentemente con la nuova pagina che bisogna caricare in memoria.

12. Consideriamo una finestra del working set di ampiezza 6 e consideriamo un programma che

accede alle proprie pagine in questa sequenza: [3, 5, 7, 2, 8, 5, 6, 5, 4, 5, 6, 7, 1, 5, 6, 4].

Supponiamo ora che sia stata appena acceduto l'ultima pagina della sequenza indicata.

Indicare qual'è il working set.

Il working set è l'insieme dei riferimenti delle pagine accedute negli ultimi 6 accessi.

Risposta: {1, 4, 5, 6, 7}

13. Che cos'è il thrashing?

È uno stato in cui un processo si può trovare quando una allocazione di frame viene gestita con sostituzione globale. Il processo sta perdendo più tempo in attività correlate alla paginazione che in attività di calcolo.

14. Nell'ambito della paginazione per la memoria virtuale, cosa si intende con

"allocazione dei frame ai processi"?

Per allocazione dei frame ai processi si indica il processo per riservare lo spazio in memoria che deve essere assegnata ai processi per rimanere in esecuzione.

15. Spiegare a cosa serve il procedimento di sostituzione delle pagine nell'ambito della

gestione della memoria virtuale.

In un sistema con memoria virtuale basata su paginazione su richiesta è importante tenere basso il numero di page fault altrimenti il tempo effettivo di accesso aumenta a dismisura e si rallenta notevolmente l'esecuzione dei processi. Per questo è cruciale scegliere bene il meccanismo di sostituzione delle pagine. Se un processo utilizza solo una parte delle pagine, la paginazione su richiesta consente di eliminare i tempi di caricamento delle pagine che non verranno utilizzate.

16. In che contesto si parla di "dirty bit" e a che cosa serve?

È un bit riservato all'interno della pagina per indicare se quest'ultima è stata modificata o

Meno.

Nell'ambito del procedimento di sostituzione delle pagine quando viene selezionata una pagina per la sostituzione, se la pagina è stata modificata quella pagina deve essere scritta sul disco.

Invece se la pagina non è stata modificata e se la copia della pagina sul disco non è stata sovrascritta (da qualche altra pagina, ad esempio), allora e possibile evitare di scrivere la pagina di memoria sul disco.

17. Spiegare cosa avviene allo scatenarsi di una page fault.

Quando avviene un page fault è perché il processo in questione non è riuscito a trovare la

pagina richiesta in memoria fisica, quindi viene sospeso e avviene una sostituzione

delle pagine, ossia viene caricata in memoria fisica da disco la pagina necessaria.

18. Spiegare cosa sono i processi "I/O bound" e quelle "CPU bound".

In un processo:

- *I/O bound I cicli di CPU burst sono molti ma molto brevi perché frequentemente interrotti da attività di I/O*
- *CPU bound I cicli di CPU burst sono meno ma più duraturi.*

19. Spiegare la differenza tra “scheduling preemptive” e “scheduling non preemptive”.

È importante la distinzione tra scheduling con diritto di prelazione (preemptive) e quello senza (non preemptive). Nel primo caso se il processo rientrato ha priorità maggiore di quello in esecuzione prima del rientro, lo spodesta. Lo scheduler entra in funzione ogni rientro nella ready queue. Nel secondo caso lo scheduler deve attendere che il processo termini o che cambi il suo stato da quello di esecuzione a quello di attesa o di pronto, a seguito, ad esempio, di una richiesta di I/O o a causa di un interrupt.

20. L’algoritmo di scheduling FIFO soffre di starvation? Perché?

No l’algoritmo di scheduling FIFO non soffre di starvation può soffrire al più di effetto convoglio.

21. Cos’è la formattazione fisica o formattazione a basso livello?

L’operazione con cui il disco viene suddiviso in settori

22. Spiegare come avviene il bootstrap nei sistemi moderni.

Il bootstrap viene effettuato in due fasi.

Nella prima viene letto in piccolissimo programma all’interno della ROM, detto bootstrap

loader, che avvia il boot che è il programma completo di avvio del Sistema Operativo che

si trova nell’area di boot del disco.

23. A cosa serve il programma fdisk?

fdisk è un programma basato su menu per la creazione e la manipolazione di tabelle di partizione.

24. A cosa serve il programma ln?

È un programma che consente di creare diversi tipi di collegamenti a file esistenti (Hard Link e Soft Link).

25. Devo implementare una delle politiche di scheduling del disco magnetico descritte a lezione, ma ancora non so quale. Che informazioni mi servono per poter implementare una qualunque di quelle politiche?

Il metodo scelto per l'allocazione dei file, La posizione in cui sono memorizzate le directory.

26. Perché non parliamo di politiche di scheduling del disco quando abbiamo a che fare con dischi a stato solido?

Perché a differenza dei dischi meccanici non ho parti in movimento quindi le prestazioni del disco sono in realtà caratterizzate solo dalla velocità di trasferimento e non anche da quella di posizionamento della testina.

27. Ipotizziamo che i cilindri di un disco magnetico siano indicizzati in maniera crescente dalla periferia verso il centro.

Supponiamo che la testina di lettura di un disco si trovi sul cilindro di indice 10. Ipotizziamo che la testina, per arrivare alla posizione attuale, si sia spostata verso il centro del disco.

Supponiamo che l'elenco delle richieste pendenti sia la seguente, ordinata in ordine di tempo di arrivo delle richieste.

12, 15, 6, 3, 6, 11, 13, 17, 20, 1.

Se uso un algoritmo di scheduling Shortest Seek Time First (SSTF), quale sara' la sequenza temporale dei prossimi accessi a disco?

Risposta: 11, 12, 13, 15, 17, 20, 6, 6, 3, 1

28. Nelle stesse ipotesi dell'esercizio precedente, quale sarà la sequenza temporale dei prossimi accessi a disco se invece uso una politica di scheduling del disco di tipo C-LOOK?

Risposta: 11, 12, 13, 15, 17, 20, 1, 3, 6, 6

29. Spiegare la differenza tra Hard link e Soft link in ambito Linux.

Gli Hard Link hanno nomi diversi ma puntano allo stesso inode, i Soft link sono un tipo speciale di file che si riferiscono ad un file (inode) diverso.

30. Cos’è un inode?

È una struttura dati sul file system che archivia e descrive attributi base su file, directory e contiene Nome, Proprietario, Dimensione, Permessi di accesso, posizione dei blocchi.

31. Che struttura si utiliza, in ambiente Linux, per memorizzare i blocchi di un file?

L'inode è una struttura che contiene gli attributi e le caratteristiche del file a cui è legato (i-node sta per information node, cioè nodo di informazione). Esiste un i-node per ogni file.