TEORIA SISTEMI OPERATIVI

Legenda: #.! = non c'è nella teoria quindi non dovrebbe chiederlo all'esame
Risposta = risposta alternativa

Domande prese dal drive

1. Cosa sono le API?
Le API (Application Program Interface) sono interfacce che vengono offerte dal SO ai programmi di alto livello per utilizzare le system calls. Le system call cambiano da SO a SO e quindi cambia il modo di implementazione delle funzioni API per adattarle al sistema operativo ma i nomi delle funzioni rimangono sempre gli stessi.

2. A cosa serve il comando strace da linea di comando?
Il comando strace serve a vedere quali system call sono usate da un programma in esecuzione.

3. Descrivi gli step del modello di compilazione per il C.
Preprocessor: processa le direttive al preprocessore (tutti i comandi preceduti dal pound #) ed elimina tutti i commenti.
Compiler: prende in input l'out del preprocessore e lo trasforma in un codice assembly.
Assembler: trasforma il codice assembly in un modulo oggetto.
Linker: prende in input i file oggetto e li collega per generare il file eseguibile.

4. Definisci i vari tipi di variabile.
Locali: ossia la sua durata di vita è pari alla durata di vita del blocco di istruzioni in cui è dichiarata
Formali: sono le variabili che vengono passate alle funzione.
Globali: sono quelle variabili che sono dichiarate fuori da tutte le funzioni. Una variabile di questo tipo può essere acceduta da tutte le funzioni di quel file e da tutte quelle in altri file in cui esiste una dichiarazione extern per la stessa variabile. Per proteggere una variabile da accessi esterni necessita della dichiarazione static.

5.! Cos'è la struttura dati 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.

6. Una stessa struttura dati ha la stessa dimensione in tutti i processori? Perché?
No, perché dipende dal processore e molto spesso i registri general purpose (di scopo generale) dei processori hanno la stessa dimensione del Bus dei Dati.

7. 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.

8. 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?
No, non è possibile perché in fase di linking viene segnalata la ridefinizione. Lascia però compilare perché preso il modulo singolarmente è valido. Però se in uno dei due moduli tale funzione è dichiarata static non è una ridefinizione in quanto static fa in modo che sia una cosa interna solo al quel modulo cioè invisibile ad altri.

9. A cosa serve l'opzione -I del preprocessore e -L e -l del compilatore?
-I (i maiuscolo) specifica il percorso dei file .h.
-L specifica il percorso delle librerie fornite dall'utente.
-l (elle minuscolo) nomelibreria specifica di utilizzare la libreria indicata dal nome ristretto nomelibreria

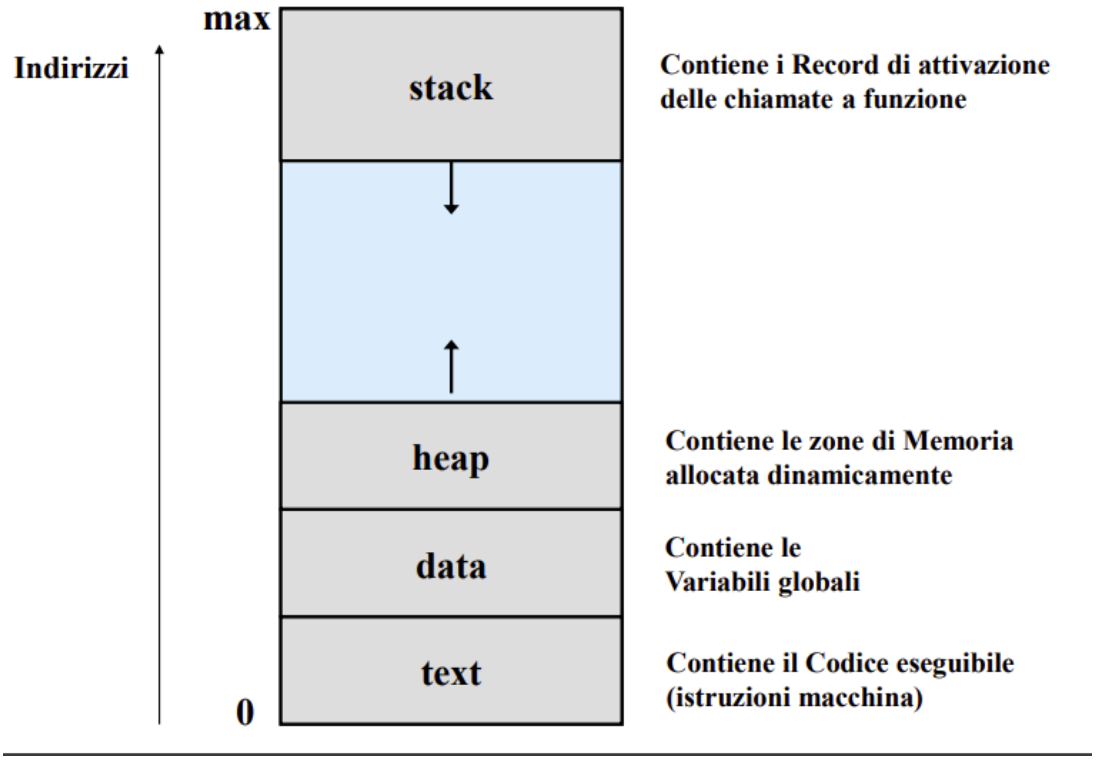
10. Differenza tra librerie statiche e dinamiche.
Librerie statiche: collegate al programma PRIMA della sua effettiva esecuzione, nella fase di preprocessing, sostituendole agli include del codice.
Librerie dinamiche: vengono collegate in due fasi. La prima controlla che si possa effettivamente collegare le librerie richieste al programma (compile time). Nella seconda (run time) un programma dedicato le carica in memoria e le collega al programma in esecuzione.

Librerie statiche:
Link-time, le librerie sono direttamente nei moduli insieme all'eseguibile generato.
Run-time, l'eseguibile caricato in memoria ha già pronto anche il codice delle librerie.

Librerie condivise:
Link-time, le librerie NON sono direttamente nei moduli insieme all'eseguibile generato.
Run-time, al momento dell'esecuzione della chiamata, viene cercata la libreria sul disco del computer, poi viene caricata in memoria e solo successivamente può essere utilizzata.

11. Che cos'è un target fittizio?
Un target fittizio, altrimenti detto .phony, è un target che non è un file e ha come scopo solo l'esecuzione di una sequenza di azioni. Bisogna indicarli tutti in fondo al Makefile ed elencarli prima come .PHONY: nomi-target.

12. Organizzazione della memoria usata dalla parte utente dei processi.



13. Che cos'è un Task?

Un Task, è il contesto in cui eseguono i processi. Ogni processo ha un proprio Task. Un Task mantiene in memoria lo stato di esecuzione di un processo, le informazioni sui segmenti di memoria utilizzati e sul valore corrente dei registri principali.

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.

14. Un Task può avere più stack a sua disposizione? Perché sì o perché no?

SI, perché ogni livello di privilegio del SO gode di un suo stack di esecuzione, in questo modo ogni livello viene mantenuto come componente atomico che non influenza gli altri. Perché ogni Task viene differenziato nei vari livelli a cui opera. (così operazioni indipendenti non interferiscono tra di loro).

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

15. Che differenza c'è tra un programma e un processo?

Un processo è l'istanziazione in memoria di un programma eseguibile (file binario eseguibile). Un programma può essere:

- un file binario eseguibile
- un processo
- una applicazione (cioè un insieme di processi)

Inoltre un processo è un'entità attiva mentre un programma è un'entità passiva

16. Quanti stack ha a disposizione un processo?

Un processo ha a disposizione uno stack per ogni livello di privilegio del kernel (4) più uno per ogni thread del processo.

17. Cos'è una System Call?

Chiamata a Sistema, avviene quando un processo fa una richiesta al kernel, in quanto una determinata operazione non sarebbe possibile; al termine dell'operazione a livello kernel il controllo viene riconsegnato al processo che ne ha fatto richiesta.
Le system call sono servizi messi a disposizione in diverse modalità. Le system call sono disponibili mediante chiamate ad interrupt effettuabili in assembly

18. Descrivi il meccanismo delle System Calls.

• un programma utente, durante l'esecuzione può invocare una system call.
• la system call è implementata nella routine di gestione di un certo interrupt n.
• la system call viene invocata eseguendo una istruzione INT n, con un certo n intero
• il programma passa i parametri all' interrupt mettendoli in registri (EAX e altri)
• chiamata ad interrupt
• switching di contesto – da modo utente a modo kernel
    • salvataggio del contesto
• le routine di gestione degli interrupt.
    • puntatori a funzione nel vettore degli interrupt.
• esecuzione su stack del kernel (per separare contesto di esecuzione)
• ritorno a modo utente
    • ripristino del contesto

19. Interrupt e Eccezioni.

Le ECCEZIONI sono sincrone rispetto alla CPU (ossia chiamate direttamente dalla CPU)
- TRAP: l'istruzione viene sospesa e portata a buon fine solo dopo la gestione del trap (*trap per debugging*)
- FAULT: l'istruzione viene interrotta, gestita l'eccezione e poi rieseguita l'istruzione dall'inizio (*Page Fault*, ossia l'impossibilità da parte di un processo di accedere ad una pagina presente nel suo spazio di indirizzo virtuale ma che non è presente nella memoria fisica. Il Sistema cerca di caricare tale pagina e poi fa ripartire il processo)
- ABORT: viene causata da un errore irrimediabile, quindi l'istruzione viene abortita e il processo killato (dividere per 0, *segmentation fault*: avviene quando un processo tenta di accedere ad un'area di memoria a cui non gli è permesso accedere quindi potrebbe danneggiare altri dati)

Gli INTERRUPT possono essere sincroni (sw) e asincroni (hw), mascherabili (possono essere rimandati) e non mascherabili (devono essere gestiti subito).

20. Spiegare la differenza tra Interrupt Sincroni e Interrupt Asincroni

Gli Interrupt Sincroni (sw) sono esplicitamente chiamati dalla CPU, mentre gli Interrupt Asincroni (hw) sono scatenati dalle periferiche che mediante il BUS avvisa la CPU che è avvenuto qualcosa da gestire (la tastiera avvisa che l'utente ha premuto un tasto).

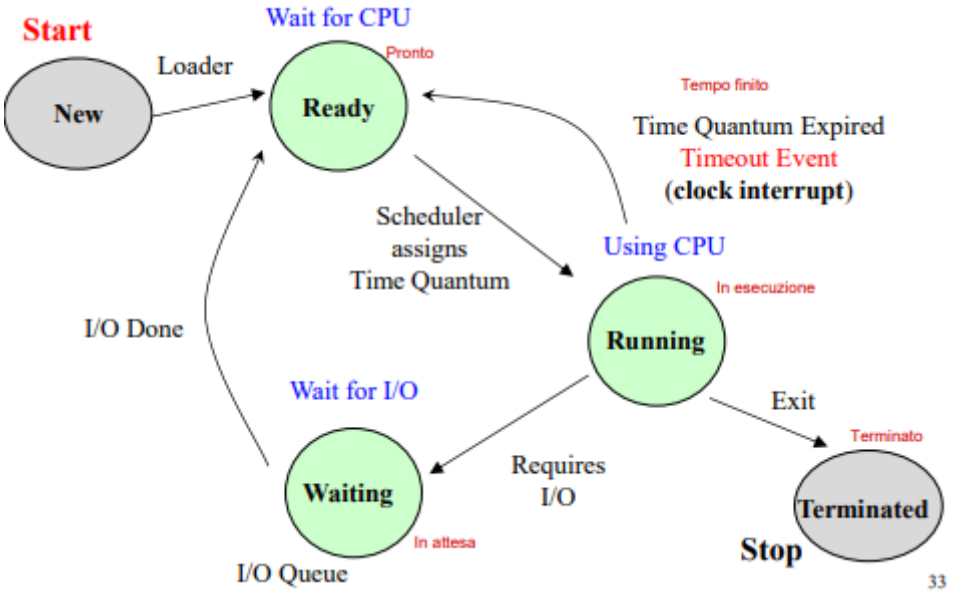
21. Disegnare il diagramma di stato dei processi, dal punto di vista dello scheduler.

Lo scheduler è quel programma che processa le richieste di accesso ad una risorsa condivisa, cioè stabilisce un ordine per l'esecuzione di tali richieste.

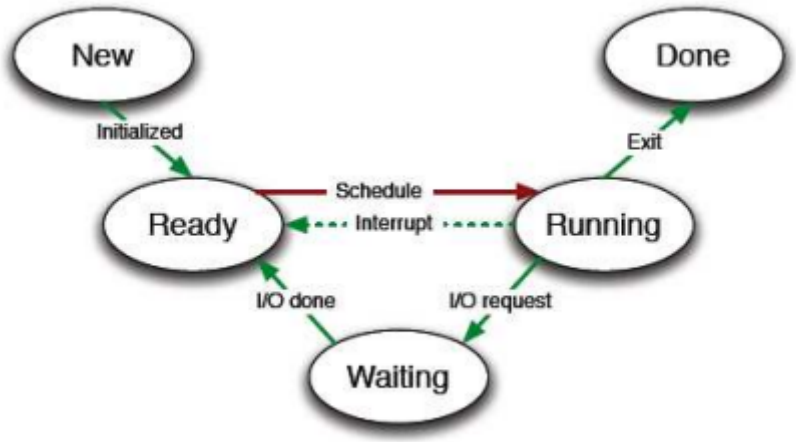
Lo SCHEDULER A LUNGO TERMINE seleziona i processi che possono entrare nella ready queue. Mentre lo SCHEDULER A BREVE TERMINE seleziona quali processi possono passare dallo stato di ready allo stato di running.

New —(loader lo carica)—> Ready(pronto per l'esecuzione)
—(esecuzione)—(exit)—>termina

N.B. durante la fase di running un processo può passare dalla fase di Waiting a causa di tempo scaduto o attesa di input/output.



oppure:



22. 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 ereditata mediante copia dalle sue subshell. Se una subshell modifica tale variabile ereditata comunque non viene modificata quella d'ambiente ma solo la copia ereditata.

23. 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*.

24. 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.

25. Che cos'è un Thread?
È un flusso di esecuzione all'interno di un processo che lo scheduler può fare eseguire separatamente e concorrentemente con il resto del processo. Ogni thread di uno stesso processo condivide le risorse del processo. Ad esempio, vede tutte le variabili globali del processo che l'ha creato.

26. Che cos'è errno?
Errno è una variabile numerica che contiene il tipo di errore che è stato provocato per ultimo/ dall'ultima istruzione.

27. Spiegare cosa si intende quando si dice 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 finché la prima istruzione non è finita. Inoltre, il risultato di quella operazione è sempre lo stesso se parte dalle stesse condizioni iniziali.

28. Cosa vuol dire che una funzione è thread safe?
Una funzione è thread safe se non causa problemi nella scrittura/lettura di strutture dati interne al so e condivise dai thread di uno stesso processo. Alcune funzioni di libreria non sono rientranti (sinonimo), per usare la forma rientrante bisogna usare la funzione nomefunzione_r.

29. Che cosa si intende con Pthread?
Un Pthread è un thread conforme allo standard POSIX, che si avvalgono di funzioni di management (creazione, eliminazione, attendere la fine), mutexes (mutua esclusione) e condition variables (sincronizzazione dipendente dal valore di variabili).
Per la creazione di un Pthread si utilizza la funzione pthread_create

30. A cosa serve la funzione pthread_join?
Serve a interrompere l'esecuzione dell'attuale processo fino a che il thread con identificativo uguale a quello passato come parametro alla funzione stessa non termina, in seguito

liberare la memoria del thread una volta che esso è terminato facendo sapere il proprio risultato.

31. Che cos'è un Pthread detached?

È un Pthread che è in grado di liberare autonomamente la memoria una volta terminato ma in questo modo non può far sapere a nessuno il proprio risultato.

32. Definire il Busy Waiting.

Con Busy Waiting ci si riferisce al continuo controllo da parte di un thread che un evento si sia verificato per permettergli di proseguire restando in stato di ready e non in wait. È da evitare in quanto comporta uno spreco di tempo e prestazioni della CPU.
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.

33. Definire il Deadlock.

Stallo o blocco. Due o più thread aspettano indefinitamente un evento che può essere causato solo da uno dei thread bloccati.
Una situazione di Deadlock avviene quando due o più processi si bloccano a vicenda.
Un esempio è rappresentato da due persone che vogliono disegnare. Per disegnare hanno a disposizione solo una riga e una matita. Per disegnare hanno bisogno di entrambe. Potendo prendere un solo oggetto per volta, se uno prende la matita e l'altro prende la riga, e se entrambi aspettano che l'altro gli dia l'oggetto che ha in mano, i due generano un Deadlock.

34. Definire la Starvation.

Con Starvation si intende l'impossibilità perpetua da parte di un processo di accedere ad una risorsa di cui necessita per essere eseguito.
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.

35. Definire la Liveness.

Per Liveness si intende la capacità di un programma 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).

36. Spiegare cosa si intende con Race Condition.

Race Condition è una situazione che si verifica in un sistema con processi e/o thread quando questo non gestisce in modo adeguato la mutua esclusione. Pertanto il risultato finale dipende esclusivamente dall'ordine in cui sono state eseguite le istruzioni.

37. Definire cosa fa la funzione cmpxchg con e senza il prefisso lock.

Tale funzione è l'equivalente di compare_and_swap in linguaggio assembly (in effetti è utilizzata nell'implementazione della funzione C compare_and_swap). Ossia se la variabile è uguale al valore che gli si vuole assegnare le si assegna il nuovo valore altrimenti nulla. Il prefisso lock garantisce che per tutta la durata dell'esecuzione dell'istruzione nessuno acceda al bus per modificare i registri che servono per la comparazione (mutua esclusione).
Risposta dettagliata:

Lock cmpxchg vuole due argomenti espliciti ed uno implicito. Il primo argomento esplicito è l'indirizzo di una variabile intera in memoria, il secondo argomento esplicito è il nome di un registro in cui c'è già il valore da assegnare alla variabile. L'argomento implicito è il registro eax in cui deve essere posto il valore da confrontare col valore della variabile. L'istruzione Lock cmpxchg preleva il valore della variabile puntata dall'indirizzo, lo confronta col valore contenuto nel registro eax, se i due valori sono uguali assegna alla variabile il nuovo valore (contenuto nel registro passato come secondo argomento). In pratica, atomicamente, se la variabile ha il valore cercato allora le viene assegnato il nuovo valore, altrimenti mantiene il suo vecchio valore.

38.! Che cos'è la Futex?
La Futex (Fast User Mutex) è una forma base di mutua esclusione e non è conforme a nessuno standard POSIX.

39. Quali sono i possibili formati di un file eseguibile in ambiente Unix/Linux e Windows?
In ambiente Unix/Linux sono a.out (vecchio) e ELF (di default sui sistemi Linux). Su Windows è PE (Portable Executable).

40. Che cos'è il PCB (Process Control Block)?
È una tabella dove vengono mantenute le informazioni sul processo e sul suo stato.
Ogni PCB si riferisce ad un unico processo ed è memorizzato in una struttura dati del kernel detta TABELLA DEI PROCESSI.

41. Che cos'è la Tabella dei Processi?
È una struttura dati con cui il kernel tiene traccia di tutti i processi attivi nel sistema. Ogni entry della tabella è il PCB di un processo attivo. Il kernel mantiene anche aggiornato un puntatore current, che punta al descritto del processo correntemente in esecuzione.

42. Differenza tra multi-threading e multi-processing?
La differenza è che i thread di uno stesso processo condividono la stessa area di memoria. Mentre i processi figli hanno ognuno un proprio spazio di indirizzamento di memoria (ossia non c'è condivisione di memoria tra i processi).

43.! Come vengono creati i processi figli?
Un processo figlio viene creato dal processo padre mediante la funzione fork(). Il processo figlio ha uno spazio di memoria che è una copia esatta di quello del padre.

44.! Quando un processo esegue una fork, le sue pagine in memoria vengono immediatamente duplicate?
No, perché non ha senso fare n copie della stesso processo padre, viene semplicemente mantenuto un riferimento alle pagine del processo padre. Le sue pagine verranno duplicate nel momento in cui verrà effettivamente fatta una modifica da uno dei due processi.

45. Che cosa sono i processi Zombie?
I processi Zombie sono quei processi che pur essendo terminati continuano ad occupare memoria inutilmente. È per questo che il processo padre una volta che i processi figli terminano deve pensare a deallocare totalmente la memoria da loro occupata, ossia ad

ucciderli, facendo una wait. Se il processo padre termina senza eseguire la wait allora i suoi processi figli si dicono Orfani. In tal caso il processo orfano viene adottato dal processo init (quello che ha originato tutti i processi al boot) che ogni tanto effettua delle wait.

46.! Che cosa si intende con IPC (Inter-Process Communication)?
(Descrivere concisamente qualche tipo di strumento messo a disposizione dallo standard POSIX per sincronizzare tra loro i processi.)
Sono meccanismi di comunicazione di cui necessitano i processi per scambiarsi dati e sincronizzazioni. Esistono due tipi di IPC:
- a memoria condivisa: ossia forniscono ai cooperanti una porzione di memoria in comune in cui possono scrivere e leggere in modo da passassi dati.
- a scambio di messaggi: forniscono ai processi la possibilità di inviare e ricevere messaggi con cui scambiarsi i dati di interesse (a distanza).

47. Che cos'è il Binding Address o rilocazione?
È l'associazione di indirizzi di memoria fisica ai dati e alle istruzione di un programma (indirizzi virtuali).
Può avvenire:
- durante la compilazione, allora gli indirizzi rimarranno invariati ad ogni esecuzione e parliamo di codice assoluto
- durante il caricamento, cioè è il loader a aggiornare tutti i riferimenti coerentemente al punto iniziale di caricamento (codice rilocabile)
- durante l'esecuzione, allora l'indirizzo di memoria effettivo viene calcolato durante l'esecuzione dalla MMU che prende in input un indirizzo logico e restituisce l'indirizzo fisico corrispondente.

48. Che cosa si intende per "allocazione della memoria" e che tipi di allocazione esistono? Descrivili.
L'allocazione della memoria consiste nell'assegnare una spazio di memoria fisica ad un programma. L'allocazione può essere:
- CONTIGUA: se tutto lo spazio assegnato ad un programma deve essere formato da celle consecutive.
- NON CONTIGUA: se è possibile assegnare aree di memoria separate.
- STATICA: ossia il programma deve mantenere l'area di memoria che gli è stata assegnata per tutta la durata della sua esecuzione.
- DINAMICA: quando durante l'esecuzione un programma può essere spostato all'interno della memoria.
- A PARTIZIONI FISSE: ossia, la memoria viene divisa in partizione a dimensione prefissata e un processo viene allocato in una partizione libera sufficientemente grande. Questo può causare frammentazione interna (quando un processo occupa una dimensione inferiore a quella della partizione).
- A PARTIZIONI DINAMICHE: la memoria non viene suddivisa ma viene assegnata ai processi che ne fanno richiesta. Questo però causa frammentazione esterna (ossia dopo un certo numero di allocazione e deallocazioni lo spazio libero appare suddiviso in piccole aree). Per questo esiste la COMPATTAZIONE, che significa spostare tutti i programmi in modo da riunire tutte le aree inutilizzate.

49.! Cos'è e a cosa serve la "mappa di bit"?
Serve ad identificare le unità di allocazione della memoria, ad ogni unità corrisponde un bit nella bitmap. 0 se l'unità è libera 1 se occupata. Un altro modo per identificare quali unità sono libere o meno è con la LISTA DI PUNTATORI, dove ogni elemento della lista specifica se si tratta di un processo (P) o di un blocco libero (H) e l'inizio e la fine del segmento.

50. Definisci i vari metodi di selezione dei blocchi liberi.
First Fit scorre la lista dei blocchi liberi dall'inizio fino a quando non trova il primo segmento vuoto grande abbastanza da contenere il processo.
Next Fit è come First Fit ma al posto che ripartire dall'inizio parte dal punto dove si era fermato all'ultima allocazione.
Best Fit, seleziona il più piccolo fra i blocchi liberi.
Worst Fit, seleziona il più grande fra i blocchi liberi.

51.! Nell'ambito della paginazione per la memoria virtuale, cosa si intende con "allocazione dei frame ai processi"?
Per "allocazione dei frame" ai processi si intende il processo di transizione da pagine a frame, ossia il passaggio delle pagine dalla memoria virtuale ai corrispondenti frame della memoria fisica.

52. Descrivi il metodo della allocazione dinamica attraverso l'uso della paginazione.
Con la paginazione la memoria virtuale (ossia lo spazio di indirizzamento logico) viene divisa in blocchi di dimensione prefissata chiamati pagine, mentre la memoria fisica viene suddivisa in blocchi della stessa dimensione delle pagine chiamati frame.
Quando un processo deve essere eseguito vengono reperiti un numero sufficiente di frame per contenere le pagine del processo.

53. Come si sceglie la dimensione delle pagine?
La dimensione delle pagine deve essere una potenza di 2, per facilitare il passaggio da indirizzi virtuali (logici) a indirizzi fisici. In genere è di 4 KB.

54. Come funziona l'MMU nella paginazione?
L'MMU prende in input un indirizzo logico formato da una coppia di valori del tipo <pagina, offset> e restituisce in indirizzo fisico formato dalla coppia <frame, offset>, avvalendosi della tabella delle pagine che si trova in memoria.

55. Che cos'è un Translation Lookaside Buffer (TLB)?
Un TLB è un insieme di registri formati da chiave (l'indice della pagina) e valore (l'elemento della tabella delle pagine ossia il frame corrispondente). Questi registri vengono utilizzati per velocizzare il processo di trasformazione di un indirizzo logico ad un indirizzo fisico.
Se la chiave è presente nel registro allora si restituisce il valore corrispondente altrimenti lo si va a cercare nella tabella della pagine in memoria.
N.B. Accedere alla memoria costa molto di più, la TLB agisce come memoria cache per la tabella delle pagine.

56. Come viene divisa la memoria in un sistema con segmentazione?
La memoria viene suddivisa in aree differenti dal punto di vista funzionale. Un segmento è un'area di memoria contenente informazioni omogenee e il suo riferimento in memoria è dato da una coppia (nome_segmento, offset). A differenza della paginazione la divisione in segmenti del programma spetta al programmatore.

57. Come funziona l'MMU nella segmentazione?
L'MMU prende in input un indirizzo logico formato da una coppia (segmento, offset), ricerca nella tabella dei segmenti l'indirizzo di inizio del segmento e somma l'offset restituendo l'indirizzo fisico corrispondente.

58. Spiega brevemente la segmentazione paginata.
Ogni segmento viene suddiviso in pagine che vengono allocate in frame liberi della memoria non necessariamente contigui.
L'indirizzo logico viene trasformato in un indirizzo virtuale e successivamente nel reale indirizzo fisico.

59. Spiegare a cosa serve il procedimento di sostituzione delle pagine nell'ambito della gestione della memoria virtuale.
Il procedimento di sostituzione delle pagine serve ad avere a disposizione sempre le pagine in memoria virtuale che servono ai processi in esecuzione, per questo si utilizzano vari algoritmi di avvicendamento.

60. A cosa serve la memoria virtuale?
È la tecnica che permette l'esecuzione di processi che non sono completamente in memoria. Permette di eseguire in concorrenza processi che nel loro complesse hanno necessità di memoria maggiore di quella disponibile.

61. Su cosa possono essere mappati gli indirizzi virtuali?
Gli indirizzi virtuali possono essere mappati sia su indirizzi fisici della memoria principale sia su memoria secondaria. In quest'ultimo caso, i dati associati in memoria secondaria vengono trasferiti in memoria principale, se non c'è più spazio si spostano in memoria secondaria i dati contenuti in memoria principale che sono considerati meno utili.

62. Che cos'è la "demand paging"?
La demand paging è la paginazione su richiesta che avviene quando un processo tenta di accedere ad una pagina in memoria principale ma non è presente e questo genera un page fault che viene gestito dal pager che si occupa di mettere in memoria principale (swap-in) la pagina mancate e aggiornare la tabella delle pagine cambiando il bit di validità.

63. Che cos'è la swap area?
La swap area è l'area del disco utilizzata per ospitare le pagine in memoria secondaria non utilizzate al momento dai processi.

64. Che cosa sono gli algoritmi di rimpiazzamento o avvicendamento?
Sono algoritmi che vengono utilizzati per gestire la sostituzione delle pagine in memoria. Ovvero per scegliere la pagina "vittima" da rimpiazzare (swap-out).

Sono algoritmi utilizzati per selezionare la pagina da sostituire.

65. Che cosa avviene con la virtualizzazione della memoria?

Con la virtualizzazione della memoria non tutto il programma è nella memoria fisica (RAM) della macchina. Per tale procedure viene virtualizzato una spazio della memoria fisica (principale) maggiore di quello fisicamente disponibile. Per farlo si crea uno spazio della memoria secondaria su cui vengono mantenute parte dei processi che non sono indispensabili immediatamente al processo.

66.! Che cos'è la mappatura della memoria?

La mappatura della memoria consente al sistema di allocare la memoria condivisa una sola volta e ai processi di vedere librerie caricate dinamicamente e memoria condivisa come parte del proprio spazio degli indirizzi inserendo nello "spazio vuoto" tra stack e heap dei puntatori a tale area di memoria condivisa.

67 Che cos'è il bit di validità?

Il bit di validità è un bit presente nella tabella delle pagine che assume 1 se il frame corrispondente è presente in memoria fisica e allora l'indirizzo logico viene trasformato in fisico, se il bit è 0 allora si genera un page fault e quindi si procede a caricare in memoria la pagina mancante.

Nella tabella delle pagine si utilizza un bit (v, per valid) che indica se la pagina è presente in memoria centrale oppure no.

68. Che cosa si intende con paginazione su richiesta pura?

È uno schema di esecuzione che avviene quando si avvia un processo senza alcuna pagina in memoria. Quindi alla prima istruzione si genererà immediatamente un page fault.

69.! Che cos'è il Backing store?

È una parte del disco (memoria secondaria) che viene utilizzata per la memorizzazione stabile delle pagine. È un sinonimo di swap area.

70. Quali sono gli effetti positivi della paginazione su richiesta?

Se un processo utilizza solo una parte delle pagine, tale procedure consente di eliminare i tempi di caricamento di pagine che non verranno mai utilizzate.

71.! Che cos'è la "copy on write"?

Quando si genera un processo figlio mediante la chiamata alla funzione fork(), le pagine del padre andrebbero teoricamente copiate nell'ambiente di esecuzione del figlio. Tale procedimento potrebbe risultare inutile in quanto se il figlio cambia immediatamente il suo ambiente di esecuzione tutte le pagine copiate diventerebbero inutili. Per evitare questo spreco di tempo, si utilizza la tecnica della copiatura su scrittura. Ossia le pagine del padre vengono inizialmente condivise e vengono effettivamente duplicate solo quando il padre o il figlio necessitano di modificare qualcosa.

72. 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. Gli algoritmi di sostituzione delle pagine prediligeranno una pagina che non è stata modificata da scartare piuttosto che una modificata in quanto il tempo di sostituzione si dimezza non dovendo modificare l'immagine su disco della pagina modificata.

73.! Elenca e spiega i vari tipi di algoritmi di sostituzione delle pagine.
FIFO: nel momento in cui si necessità di uno swap-out, si sceglie la pagina con l'istante di caricamento più basso (la più vecchia)
LRU (Least Recently Used): scarica la pagina che non viene acceduta da più tempo, utilizzando un bit di riferimento (0 se la pagina non è stata acceduta, 1 altrimenti) oppure un contatore degli accessi.
Bit supplementari di riferimento: ogni N millisecondi viene salvato il bit di riferimento di tutte le pagine (si usa un byte per ogni pagina) e si rialzeranno tutti i bit. Nel momento dello swap-out la pagina usata meno recentemente è quella con il byte di valore più basso.
Seconda Chance: utilizza un solo bit per pagina. Le pagine sono tenute in una coda circolare con un puntatore: se trova un bit a 0 sostituisce se lo trova a 1 lo azzera. In questo modo la pagina con bit 1 ha una seconda chance.
Seconda Chance migliorato: usa la coppia (bit di riferimento, dirty bit). Privilegia lo scaricamento di pagine non modificate per rendere più veloce lo swap-out.

74.! Che cos'è l'anomalia di Belady?
È ciò di cui può soffrire un algoritmo nel caso in cui aumentando il numero di frame disponibili al posto che diminuire, i page fault aumentano.

75.! Spiegare come dovrebbe funzionare "l'algoritmo ottimale" di sostituzione delle pagine.
Il problema della sostituzione delle pagine mira a minimizzare il numero dei page fault.
L'algoritmo ottimale è quello che conoscendo la sequenza delle pagine richieste le fa trovare sempre al momento giusto in memoria centrale (predice il futurooooo).
L'algoritmo ottimale è quello che rende 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.

76.! Spiega la differenza tra sostituzione globale e locale.
Si parla di sostituzione globale quando la richiesta di sostituzione è soddisfatta prelevando un frame da un processo qualunque. Locale invece quando viene prelevata dal processo che ha fatto la richiesta.
Nel caso di una sostituzione globale può accadere che un processo con priorità alta porti un processo con priorità inferiore ad un numero di frame inferiore al minimo. Occorre in questo caso richiamare lo scheduling e rimuovere il processo dalla ready queue prima che vada in trashing.

77.! Che cos'è il thrashing?
Il thrashing si verifica quando un algoritmo di sostituzione delle pagine decide di scartare una pagina che verrà usata a breve. Quindi vuol dire che il processo si trova a dedicare più tempo alla sostituzione delle pagine che alla effettiva esecuzione di istruzioni. Questo può causare un circolo vizioso in quanto un processo in thrashing non usa CPU, il SO rileva un basso uso della CPU e quindi carica altri processi, questo porta altri processi ad andare in thrashing.

78.! 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: {7, 1, 6, 5, 4}

79.! Che cos'è il working set?
È l'insieme delle pagine che contengono gli ultimi x riferimenti.

80.! Che cos'è e a cosa serve un file mappato in memoria?
Un file mappato è un file creato su disco con un riferimento ad un'area di memoria. I processi accedono in modo indiretto a tale area grazie al file mappato su disco a cui possono accedere tutti. È usato come salvataggio dell'area di memoria.

81.! Che cosa sono i registri STLR (Segment Table Limit Register) e STBR (Segment Table Base Register)?
STBR contiene l'indirizzo di inizio della tabella dei segmenti del processo in esecuzione.
STLR contiene la dimensione della tabella.

82. Quali indirizzi vengono specificati dalle istruzioni di un programma in esecuzione?
Virtuali.

83. Quali indirizzi vengono spediti sul bus degli indirizzi?
Fisici

84. Se debuggo un programma che usa puntatori, che indirizzi vedo nelle variabili puntatori?
Virtuali.

85.! A cosa serve il comando sysctl -w kernel.randomize_va_space = 0?
Serve a far sì che il kernel non randomizzi l'indirizzamento fisico perciò ad esempio nel lanciare uno stesso programma più volte che stampa un indirizzo avrò sempre lo stesso output.

86.! Che cos'è un Exploit e a cosa serve l'Address Space Layout Randomization?
Un Exploit è un codice che viene scritto per ottenere privilegi che non ti appartengono sfruttando una vulnerabilità. L'ASLR è la tecnica tramite la quale il sistema operativo, quando carica un programma da eseguire, per collocare i segmenti in memoria aggiunge una quantità casuale agli indirizzi base.

87.! In linguaggio assembly, cos'è un'istruzione di salto e quali tipi di istruzione di salto possiamo trovare?
Un'istruzione di salto (JMP) va a modificare il registro CS (Code Segment) che indica l'istruzione successiva. Quindi l'istruzione che verrà eseguita dopo il JMP non sarà quella scritta dopo ma quella indicata dalla label specificata.
Possiamo avere istruzioni di salto:

- Condizionale: se la condizione è vera salta all'indirizzo specificato, altrimenti prosegue con l'istruzione che nell'eseguibile si trova dopo.
- Non condizionale: senza alcuna condizione.
- Assoluto: ossia viene specificato di andare all'indirizzo x (JMP 1000, JNE ZZ).
- Relativo: viene indicato a quante istruzioni di distanza si trova l'istruzione dove bisogna saltare da quella attuale (JRE 250)

88.! Spiegare cosa sono i processi "I/O bound" e quelli "CPU bound".
I processi "I/O bound" sono caratterizzati da un'alta richiesta di input/output, ossia impiegano più tempo a richiedere dati (attività di I/O burst) che ad elaborarli. Un processo, invece, "CPU bound" è caratterizzato da un altissimo consumo di CPU (attività di CPU burst) rispetto all'I/O.

89.! Che cos'è una ready queue?
È la struttura dati in cui lo scheduler mantiene i PCB dei processi pronti ad essere eseguiti.

90.! Spiegare la differenza tra "scheduling preemptive" e "scheduling non preemptive".
Le due situazioni dipendono da come è implementato lo scheduler. Uno scheduler di tipo preemptive nel momento in cui un processo entra in ready se ha una priorità maggiore rispetto a quello in esecuzione, quest'ultimo viene interrotto per far partire quello appena entrato. Uno scheduler di tipo non preemptive, invece, anche se il processo che entra in ready ha priorità maggiore esso dovrà aspettare la terminazione di quello in esecuzione.

91.! Che cos'è il dispatcher?
È una parte del SO che passa effettivamente il controllo della CPU al processo selezionato dallo scheduler a breve termine.

92.! Che cos'è il Throughput (produttività) e il tempo di turnaround?
Il throughput è il numero di processi completati nell'unità di tempo.
Il tempo di turnaround (tempo di completamento) è il tempo necessario per eseguire un determinato processo.

93.! Descrivere i vari algoritmi di scheduling.
FCFS/FIFO: quando un processo entra in coda viene inserito in fondo
SJF (Shortest Job First): seleziona il prossimo processo prelevando quello con i tempi di esecuzione minori (può generare starvation).
Priority Scheduling: ad ogni processo viene assegnato un numero che rappresenta la sua priorità, quindi verranno eseguiti prima i processi con priorità maggiore.
Round Robin: utilizza il metodo FCFS, però ogni volta che un processo viene messo in esecuzione lo scheduler imposta un timer, scaduto il quale il processo viene interrotto e reinserito nella ready queue per dare l'impressione di una esecuzione simultanea di tutti i processi.

94.! Che cos'è il convoy effect?
È ciò che può avvenire in uno scheduling di tipo FCFS, ovvero che un processo molto lungo può bloccare molti processi che avrebbero eseguito molto più velocemente aumentando così i tempi medi di attesa.

95.! È possibile adottare politiche differenti di scheduling?
Si, grazie alle code multiple. In ogni coda può essere implementata una politica di scheduling differente.

96.! Come sono fatti i dischi meccanici (magnetici) ?
Sono composti da un insieme di piatti. Ogni piatto è diviso in tracce circolari che a loro volta sono divise in settori. L'insieme delle tracce equidistanti dal centro si dice cilindro.
Per accedere ad un settore bisogna specificare la superficie, la traccia e il settore stesso.

97.! Cos'è la formattazione fisica o formattazione a basso livello?
La formattazione fisica è essenzialmente la preparazione del disco, da vergine viene suddiviso con questo tipo di formattazione, in sezioni e per ogni sezione viene predisposta un'area per il controllo degli errori. Uno disco formattato a basso livello è utilizzabile solo dalla macchine e non dall'utente perché non è ancora stato definito un File System utilizzabile che verrà successivamente creato con una formattazione ad alto livello o logica.

98. Spiegare come avviene il bootstrap nei sistemi moderni.
Il bootstrap viene effettuato in due fasi.
Nella prima viene letto un 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 sul disco (RAM).

99.! 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 perciò posso leggere in tempo costante in ogni punto del disco in un qualunque momento.

100.! Elenca e descrivi i tipi di scheduling del disco?
- FCFS: la prima richiesta arrivata è la prima ad essere eseguita.

- SSTF (Shortest Seek Time First): vengono servite prima tutte le richieste più vicine alla posizione corrente. Soffre di Starvation, le richieste più periferiche rischiano di non essere mai soddisfatte.
- SCAN: il braccio si sposta da un estremo all'altro del disco servendo tutte le richieste che incontra.
- C-SCAN: opera come SCAN ma non inverte mai la direzione, ovvero una volta arrivato a fine corsa il braccio riparte dalla parte opposta (traccia 0).
- C-LOOK: come C-SCAN ma il bracci non apriva a fine corsa ma si ferma all'ultima richiesta, risparmiando tempo inutile per arrivare all'ultima traccia del disco.

101.! 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 sarà la sequenza temporale dei prossimi accessi a disco?
Risposta: 11, 12, 13, 15, 17, 20, 6, 6, 3, 1
Accendo nella posizione più vicina a quella attuale (algoritmo greedy).

102.! 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
Continuo nella direzione corrente fino a quando non incontro l'ultima richiesta in quella direzione (cioè che non provoca inversione) e vado alla richiesta più lontana nell'altro verso e torno indietro.

103.! A cosa serve il programma ln?
È un programma che consente di creare collegamenti simbolici (in windows la frecciata sulle icone, se cambio ad esempio la posizione di un file/directory il collegamento non è più valido) o fisici (ciò che collega una directory al suo contenuto, se cambia il contenuto il collegamento rimane valido) a file e directory.

104.! A cosa serve il programma fdisk?
È un'utility a linea di comando per la gestione dei dischi per creare e manipolare le tabelle delle partizioni e le partizioni stesse (ridimensionamento, modifica ed eliminazione).

Domande dei pdf YYY01 e YYY02

105. 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 di inizio della stringa str. Dopo l'assegnamento, l'indirizzo contenuto nella variabile p specifica un indirizzo 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 (MMU).

106. 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.

107. 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

108. Che cosa accade durante il 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.

109.! 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.

110. 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 un'unica operazione del bus tra CPU e memorie.

111.! 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.

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

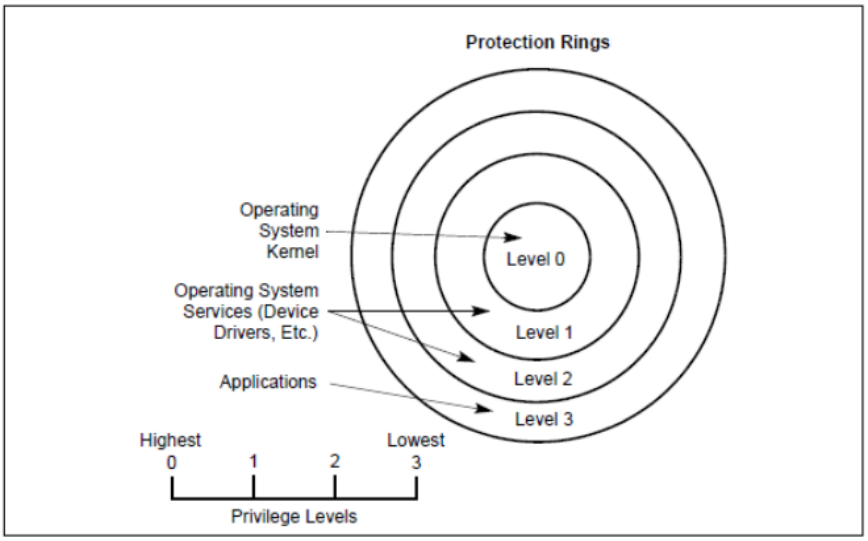
L’opzione -Wl,-rpath,RunTimeLibDir serve per indicare la posizione delle librerie dinamiche a runtime.

113. Cos’è il modo Kernel?

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

114. 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.



I livelli di privilegio della CPU, noti anche come “rings”, sono un meccanismo di sicurezza utilizzato per proteggere il sistema operativo e le applicazioni dalle eventuali operazioni non autorizzate. Ci sono generalmente quattro livelli di privilegio, noti come anello 0, anello 1, anello 2 e anello 3. L'anello più elevato, l'anello 0, è riservato al sistema operativo e alle applicazioni di alto livello (kernel), mentre l'anello più basso, l'anello 3, è utilizzato per le applicazioni e i programmi di livello inferiore. Ciò consente di limitare le operazioni che un programma o un'applicazione può eseguire, prevenendo così l'accesso non autorizzato a risorse critiche del sistema.

115. 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 variabili globali.

116. 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.

117. 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

118. 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.

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

Tutti i thread di uno stesso processo condividono le risorse del processo (dati globali e CPU) ed eseguono nello stesso spazio utente.

120. 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().

121. Se un processo possiede diversi posix thread e quel processo genere un processo figlio, i thread del processo figlio "vedono" le variabili 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.

122. 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.

123. 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

124. 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);
}
fa_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.

125. 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)))

126. 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.

127. 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.

128. Che struttura si utilizza, 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.

129. Quali sono i metodi di accesso ai file?
Il SO può fornire diversi metodi di accesso ai file:
– Sequenziale: le informazioni del file vengono elaborate in ordine, un record dopo l'altro.
    • Nastri
– Diretto: il file è costituito da un insieme ordinato di blocchi a cui si accede direttamente.
    • Dischi
– Attraverso indice: al file vero e proprio è associato un file indice con lo scopo di velocizzare le ricerche.
    • Database

130. Quali sono i tipi di allocazione dei blocchi?
L'allocazione dei blocchi può avvenire in tre metodologie:
- Contiguous allocation, ogni file occupa un insieme di blocchi contigui del disco.
- Linked allocation, evita i problemi di frammentazione della contigua.
- Indexed allocation, evita i problemi di accesso causati dalla concatenata.