SLIDE CARPENTIERI 2020-21

Lezione 1 (Cap.1)

Un sistema operativo è un insieme di programmi che funge da intermediario tra un utente e gli elementi fisici di un calcolatore, garantendone un uso efficiente.
Serve ad eseguire programmi d'applicazione e rende più facile la risoluzione di problemi che gli utenti devono affrontare.
In un sistema di calcolo troviamo:

- Hardware, la parte che fornisce le risorse di calcolo fondamentali quali CPU e memoria;
- Sistema Operativo, che controlla e coordina l'uso dei dispositivi;
- Programmi d'Applicazione, che definiscono come sono usate le risorse per risolvere problemi;
- Utenti, cioè persone, macchine o altri calcolatori.

Un sistema operativo si compone di diverse parti:

- Assegnatore di risorse, che gestisce e assegna le risorse;
- Programma di controllo, che gestisce l'esecuzione dei programmi utenti e le operazioni I/O;
- Nucleo (o Kernel), il solo programma che funziona sempre nel calcolatore.

Un primo, rudimentale sistema operativo è il sistema mainframe,che è monitor residente: trasferisce automaticamente il controllo da un lavoro al successivo e raggruppa lavori con requisiti simili.

In un sistema operativo multiprogrammato sono sempre presenti diversi lavori nella memoria centrale, per cui la CPU non rimane mai inattiva.
Il sistema si occupa di ripartire la memoria tra i vari lavori in entrata e decidere quale eseguire prima.

In un sistema a partizione del tempo d'elaborazione la CPU viene ripartita tra i vari lavori tenuti in memoria. Ciascun utente dispone di almeno un proprio programma nella memoria e comunica direttamente con il sistema avendo l'impressione di disporre dell'intero calcolatore, quando in realtà il sistema passa rapidamente da una parte all'altra.

I sistemi paralleli sono dotati di più unità di elaborazioni, cioè con più CPU in stretta collaborazione.
I processori condividono la memoria e i temporizzatori dei cicli macchina; la comunicazione avviene attraverso la memoria condivisa. Sono maggiormente produttivi ed affidabili.
La multielaborazione può avvenire sia in modo simmetrico (SMP) che asimmetrico (AMP):

- nel primo ogni unità esegue una copia del sistema operativo, consentendo di eseguire molti processi contemporaneamente;
- nel secondo viene assegnato un compito specifico a ciascuna unità da una unità principale.

I sistemi distribuiti si basano sulle reti per realizzare le loro funzioni e sfruttano le capacità di comunicazione. Ciasuna unità ha la propria memoria locale e comunica con le altre per tramite diversi mezzi. Richiedono un'infrastruttura di rete LAN o WAN e possono essere sia server-client che peer-to-peer.

Una batteria di sistemi o cluster system è basata sull'uso congiunto di più unità di elaborazione riunite e fornisce elevatà disponibilità. Possono sia essere asimmetriche, in cui un calcolatore rimane in attesa attiva e l'altro esegue le applicazioni, oppure simmetriche, in cui ogni calcolatore elabora e vigila.

I sistemi di elaborazione in tempo reale sono usati nella gestione di dispositivi di controllo per applicazioni specifiche e presentano vincoli di tempo fissati e ben definiti entro i quali si deve effettuare l'elaborazione.
Possono essere di due tipi: stretto (hard real-time) oppure debole (soft real-time).

I sistemi palmari sono usati per i telefoni cellulari e viste le dimensioni ridotte questi dispongono di memoria limitata ed unità di elaborazione lente.

Lezione 2 (Cap.2)

Un moderno calcolatore d’uso generale è composto da una CPU e da un certo numero di controllori di dispositivi, che sono connessi attraverso un canale di comunicazione comune (bus) che permette l’accesso alla memoria condivisa dal sistema.
Ciascun controllore ha un buffer locale e si occupa di un particolare dispositivo fisico.
La CPU sposta i dati tra la memoria principale e i buffer locali e riceve un segnale d'interruzione quando un'operazione termina.
L'architettura di gestione delle interruzioni deve salvare l'indirizzo dell'istruzione interrotta e trasferire il controllo all'appropriata procedura di servizio.
Un segnale d'eccezione (trap) può essere causato da un evento eccezionale in un programma in esecuzione oppure da una richiesta di un programma utente.

Un moderno sistema operativo è detto guidato dalle interruzioni o 'interrupt driven'. Il sistema operativo memorizza l'indirizzo di ritorno nello stack di sistema. Una volta determinato il tipo di interruzione ('polling' oppure 'vectored interrupt system'), diversi segmenti di codice determinano le azioni da intraprendere.

Per la struttura di I/O possono essere seguite due strade: sincrona e asincrona.
Una volta iniziato il processo di I/O, si restituisce il controllo al processo utente solo dopo il completamento dell'operazione di I/O: l'istruzione <wait> sospende la CPU, altrimenti, in assenza di tale istruzione, si può generare un ciclo di attesa.
Nell'asincrono il controllo viene restituito immediatamente al

processo utente, senza attendere il completamento dell'operazione I/O: tramite system call il programma utente chiede di attendere il completamento dell'operazione al sistema operativo, il quale individua il controllore del dispositivo che ha emesso il segnale, accede alla tabella di stato dei dispositivi e modifica l'elemento indicando l'occorrenza dell'interruzione.

Per dispositivi di I/O veloci viene usata la tecnica di accesso diretto alla memoria: il controllore trasferisce un intero blocco di dati dalla propria memoria a quella centrale, o viceversa, senza interventi della CPU e richiedendo una sola interruzione per blocco.

La memoria di un calcolatore può essere divisa in centrale, direttamente accessibile alla CPU, secondaria, che estende la centrale e contiene gran quantità di dati in modo permanente, e disco magnetico, un piatto rigido di metallo o vetro, coperto da materiale magnetico e diviso in tracce e settori.
I componenti della memoria si possono organizzare gerarchicamente in base a velocità, costo e volatilità:
registri -> memoria centrale -> disco RAM -> dischi magnetici -> dischi ottici -> nastri magnetici

Tra registri e memoria centrale si inserisce poi la memoria cache, una memoria ad alta velocità in cui si registrano dati recentemente usati e che si prevede saranno usati nuovamente a breve.
La condivisione delle risorse rende necessario che il sistema operativo garantisca che un programma malfunzionante non abbia effetti su altri programmi.

Esistono due modi di funzionamento, indicati dal bit di modo: quello d'utente (1), per cui l'istruzione corrente è eseguita per un utente, e quello di sistema (0), per cui l'istruzione corrente è eseguita per il sistema, il quale possiede istruzioni privilegiate.
Tutte le istruzioni I/O sono istruzioni privilegiate, dunque è necessario evitare che l'utente possa ottenere il controllo del calcolatore quando questo è nel modo di sistema.

La protezione della memoria invece è parziale, in quanto vanno garantite quella per il vettore delle interruzioni e per le relative procedure di servizio. Viene realizzata con due registri: quello di base indica il più basso indirizzo a cui il programma può accedere, mentre quello di limite indica la dimensione dell'intervallo. Le aree al di fuori dell'intervallo sono protette.
Funzionando nel modo di sistema, il sistema operativo ha il privilegio di accedere indiscriminatamente sia alla memoria ad esso riservata che a quella riservata agli utenti, in modo da caricarvi i programmi.

Per proteggere la CPU viene usato il temporizzatore (timer), che

invia un segnale d'interruzione alla CPU periodicamente, in modo che sia sempre il sistema operativo ad avere il controllo. Sono usati inoltre per realizzare la partizione del tempo d'elaborazione e il calcolo dell'ora corrente.

Lezione 3 (Cap.3)

Un processo è un programma in esecuzione che richiede varie risorse, tra cui tempo di CPU, memoria, file e dispositivi di I/O. Il S.O. crea e cancella processi utenti e di sistema, li sospende, li ripristina e fornisce meccanismi sia per sincronizzarli che per metterli in comunicazione.

La memoria centrale è un vastissimo vettore di dati contenente parole, ciascuna dotata del proprio indirizzo, ed è condivisa da CPU e dispositivi di I/O. È velocemente accessibile e volatile, cioè perde informazioni in caso di guasti.
Il S.O. tiene traccia di quali parti sono occupate e da cosa, decide quali processi caricarvi e assegna e revoca spazio a seconda delle necessità.

Un file è una raccolta di informazioni correlate definite da un creatore e rappresentano, comunemente, programmi (sia sorgente che oggetto) e dati.
Il S.O. crea e cancella file e directory, fornisce le funzioni fondamentali per la gestione di questi, associa file a dispositivi di memoria secondaria e crea backup su dispositivi non volatili.

Il sistema di I/O è composto da un sistema buffer-caching, i driver per gli specifici dispositivi ed un'interfaccia generale per essi.

La memoria secondaria, generalmente implementata tramite dischi, supporta la memoria centrale. Il S.O. gestisce e assegna lo spazio e si occupa dello scheduling del disco.

Un sistema distribuito è un insieme di unità d'elaborazione distinte collegate tramite una rete di comunicazione, la quale avviene secondo un protocollo.
Essi offrono all'utente varie risorse in modo da accellerare i calcoli, aumentare la disponibilità dei dati e incrementare l'affidabilità.
La protezione è definita da ogni meccanismo che controlla l'accesso da parte di programmi, processi o utenti alle risorse di un sistema di calcolo.
Esso deve distinguere tra uso autorizzato e non, specificare i controlli che devono essere attivati e migliorare l'affidabilità.
Il programma che legge e interpreta le istruzioni di controllo prende il nome di interprete di schede di controllo o di riga di comando o ancora 'shell'. La sua funzione consiste nel prelevare ed eseguire le istruzioni di comando.

Il sistema operativo mette a disposizione una serie di funzionalità, tra cui:

- Esecuzione di un programma: capacità di caricare ed eseguire un programma;
- Operazioni di I/O: poiché i programmi utenti non possono eseguirli direttamente;
- Gestione del file system: capacità di creare, scrivere, leggere e cancellare file;
- Comunicazioni: scambio di informazioni sia tramite memoria condivisa che scambio di messaggi;
- Rilevamento di errori: capacità di rilevare eventuali errori in CPU, dispositivi e programmi;
- Assegnazione delle risorse: provvede ad assegnare le risorse necessarie a programmi eseguiti in contemporanea;
- Contabilizzazione dell'uso delle risorse: registrazione di quante e quali risorse vengono usate da ciascun utente;
- Protezione: assicura che l'accesso alle risorse del sistema sia controllato.

Le chiamate del sistema costituiscono l'interfaccia tra un processo e il S.O. In genere sono disponibili in forma di istruzioni in linguaggio assemblativo ed alcuni programmi scritti in linguaggi di alto livello possono eseguirle.

Si possono passare parametri al S.O. tramite tre metodi diversi:
- Passare i parametri in registri;
- Memorizzare i parametri in una tabella o blocco e passare l'indirizzo ad essa;
- Collocare i parametri in una pila da cui il sistema può prelevarli.

Per la maggior parte degli utenti, l'interfaccia col S.O. è definita dai programmi di sistema piuttosto che da chiamate.

Il **MS-DOS** è progettato per fornire la massima funzionalità nel minimo spazio, non è diviso in moduli, ha una struttura semplice e non separa bene interfacce e livelli di funzionalità.

Lo **UNIX** è un esempio di strutturazione inizialmente limitata dalle funzioni dell'architettura sottostante. È formato da programmi di sistema e nucleo, il quale è tutto ciò che si trova tra l'interfaccia delle chiamate di sistema e i dispositivi fisici. Combina molteplici funzioni in un singolo livello: fornisce il file system, lo scheduling della CPU e la gestione della memoria.

Con il metodo stratificato il S.O. è diviso in livelli, o strati, in cui lo strato fisico corrisponde al livello 0 e l'interfaccia utente al livello n.
Secondo l'orientamento a micronucleo, il S.O. viene progettato rimuovendo dal nucleo tutti i componenti non essenziali, realizzandoli come programmi utente e di sistema.
Si comunica tramite scambio di messaggi e porta diversi vantaggi: è più sicuro e affidabile e consente di estendere il S.O. facilmente e adattarlo a diverse architetture.
Dal concetto di metodo stratificato si sviluppa quello di macchina virtuale:

essa è una interfaccia identica all'architettura sottostante che ha l'illusione di disporre CPU e memoria proprie; il calcolatore fisico condivide le risorse.

La partizione del tempo d'uso della CPU si può usare sia per condividere la CPU che per dare l'illusione che gli utenti dispongano di una propria; la gestione asincrona di I/O ed esecuzione di più processi consente di creare lettori di schede e stampanti virtuali.
Un normale terminale di un sistema a partizione del tempo funziona da console della macchina virtuale.
Se l'essere isolate da un lato rende le macchine virtuali sicure, in quanto proteggono le risorse, dall'altro c'è l'impossibilità di condividere in maniera diretta le risorse.
Offrono un ambiente perfetto per lo sviluppo di nuovi S.O., in quanto evitano di apportare modifiche fisiche che potrebbero portare errori altrove.

I programmi Java sono eseguiti sulla Java Virtual Machine, che consiste di un caricatore delle classi, un verificatore delle classi e un interprete del linguaggio che esegue il bytecode; in più le prestazioni sono migliorate da un compilatore istantaneo Just-In-Time.

Quando si progetta un S.O., bisogna tener presente diversi scopi: dal punto di vista dell'utente il S.O. deve essere utile, semplice da usare, affidabile ed efficiente, mentre per il sistema questo deve essere di facile progettazione, realizzazione e manutenzione ed inoltre deve essere flesibile, senza errori ed efficiente.

I meccanismi determinano come eseguire qualcosa, mentre i criteri stabiliscono il cosa e sono dunque coloro che consentono la flessibilità in quanto situazionali.
Tradizionalmente i S.O. sono scritti in linguaggio assemblativo, mentre oggi si prefisce C o C++, poiché si hanno codici più veloci da scrivere, compatti e semplici sia da capire che da adattare.

Il processo di generazione di sistemi (SYSGEN) configura o genera il sistema: il Booting indica l'avviamento del calcolatore attraverso il caricamento del nucleo, mentre il Bootstrap program è un segmento di ROM che individua, carica ed avvia il nucleo.

Lezione 4 (Cap.4)

Un sistema operativo esegue differenti programmi:

- Un sistema a lotti (batch system) esegue lavori;
- Un sistema a partizione del tempo (time-shared system) esegue programmi utenti o task.

Un processo, o lavoro, è un programma in esecuzione sequenziale che comprende:

- Contatore di programma (program counter);
- Pila (stack);
- Sezione di dati (data section).

Durante l'esecuzione è soggetto a diversi cambiamenti di stato:

- Nuovo, in cui si crea il processo;
- Esecuzione, in cui le istruzioni vengono eseguite;
- Attesa, in cui si attende il verificarsi di qualche evento;
- Pronto, in cui il processo attende di assere assegnato ad un'unità di elaborazione;
- Terminato, in cui il processo ha terminato l'esecuzione.

Il Process Control Block (PCB) è un blocco di informazioni specifiche connesse ad un processo: stato del processo, contatore di programma, registri e informazioni sullo scheduling, gestione della memoria, contabilizzazione delle risorse e stato di I/O.

Ci sono più tipi di code di scheduling dei processi: la 'job queue' è l'insieme generale, la 'ready queue' comprende quelli già caricari in memoria e in attesa di essere eseguiti e infine la 'device queue' comprende i processi che attendono la disponibilità di un particolare dispositivo di I/O.

Lo scheduler a lungo termine o 'job scheduler' seleziona i processi che devono essere caricati nella coda dei processi pronti, mentre lo scheduler di CPU o 'short-term scheduler' si occupa di selezionare un lavoro tra quelli pronti da assegnare alla CPU.
Mentre quest'ultimo viene invocato frequentemente (millisecondi) e deve dunque essere veloce, il primo può essere invece lento in quanto invocato meno frequentemente (secondi o minuti).
Lo scheduler a lungo termine controlla il grado di multiprogrammazione.

I processi possono essere divisi in I/O-bound se impiegano la maggior parte del proprio tempo nell'esecuzione di operazioni di I/O, oppure CPU-bound se si concentrano sulle elaborazioni.

Il cambio di contesto è il passaggio da un processo all'altro e la sua velocità dipende dall'architettura: durante questo lasso di tempo viene registrato lo stato del vecchio processo e caricato il nuovo.
Un processo genitore può generare numerosi processi figli e, poiché questo vale anche per loro, si possono generare alberi di processi: un genitore e un figlio possono condividere tutte, alcune o nessuna risorsa e possono sia essere svolti in contemporanea che attendere alcuni o tutti i figli.

In UNIX la chiamata di sistema <fork> crea un nuovo processo e chiamare immediatamente dopo <exec> sostituisce lo spazio di memoria del processo con un nuovo programma.

Quando un processo termina può esserci una chiamata di sistema <wait>, se alcuni dati vanno riportati al processo genitore, oppure una <exit> se il processo vuole essere cancellato subito.
Un processo genitore può inoltre porre termine all'esecuzione di un figlio (abort) quando il figlio ha ecceduto nell'uso di alcune

risorse o non è più richiesto oppure quando è il processo genitore stesso a terminare, comportando una terminazione a cascata.

Se un processo non può influire o essere influenzato da altri processi è detto indipendente, altrimenti è detto cooperante.
Per i processi cooperanti il paradigma usuale stabilisce una relazione tra produttore e consumatore per lo scambio di informazioni e si differenziano i casi in cui la dimensione del vettore è fissata oppure no.

Tramite le funzioni di IPC i processi possono comunicare tra loro, sincronizzando le proprie azioni senza dover ricorrere a dati condivisi. Vengono fornite solo due operazioni, <send> e <receive>, che potranno essere usate da due processi P e Q dopo che avranno stabilito un canale di comunicazione, fisico o logico.

Nella comunicazione diretta i processi devono nominarsi reciprocamente in modo esplicito: il canale è unico per ciascuna coppia, stabilito automaticamente e, in genere, bidirezionale.

Nella comunicazione indiretta i messaggi sono inviati a delle porte identificate in modo univoco: tra una coppia si stabilisce un canale solo se entrambi condividono una stessa porta e può essercene più di uno; il canale può essere associato a più di due processi e può essere unidirezionale o bidirezionale.
In questo caso si aggiungono operazioni per creare e rimuovere porte e si modificano <send> e <receive> per operare con esse.

Lo scambio di messaggi, così come send e receive, può essere sincrono (bloccante) o asincrono (non bloccante).
I messaggi risiedono in code temporanee realizzabili in tre versioni diverse:

- Capacità zero, in cui il trasmittente deve fermarsi finché il ricevente prende il messaggio;
- Capacità limitata, in cui viene stabilità una lunghezza n e si attende solo quando questa si riempie;
- Capacità illimitata, in cui la coda ha potenzialmente infinita ed il trasmittente non si ferma mai.

La comunicazione nei sistemi client-server avviene con diversi metodi:

- Una socket è una estremità di un canale di comunicazione e si indica come un indirizzo IP concatenato ad un numero di porta; la comunicazione avviene attraverso una coppia di socket.
- Le chiamate di procedure remote (RPC) sono usate per comunicazioni tra sistemi connessi tramite rete. Un segmento di codice (stub) individua la porta del server e struttura i parametri e questo avviene in modo analogo lato server.
- L'invocazione di metodi remoti (RMI) è una funzione Java simile alla RPC che consente a un thread di invocare un metodo di un oggetto remoto.

Lezione 5-1 (Cap.5)

I processi possono essere a thread singolo oppure a multithread, con i vantaggi di migliori tempi di risposta, condivisione delle risorse ed uso di più unità di elaborazione.

I thread al livello d'utente sono gestiti come uno strato separato sopra il nucleo del S.O. e realizzati tramite una libreria di funzioni per la creazione, lo scheduling e la gestione.
I thread al livello del nucleo sono invece gestiti dal S.O.
Il multithread può essere implementato seguendo diversi modelli:

- Nel modello da molti a uno, usato su sistemi che non li gestiscono a livello del nucleo, a molti thread al livello d'utente ne corrisponde uno al livello del nucleo.
- Nel modello uno a uno ad ogni thread al livello d'utente ne corrisponde uno a livello del nucleo.
- Nel modello molti a molti vengono messi in corrispondenza più thread al livello d'utente con al più lo stesso numero di thread al livello del nucleo, consentendo ai programmatori di crearli liberamente.

Lo standard POSIX che definisce l'API per la creazione e la sincronizzazione dei thread prende il nome di Pthread. Non sono realizzazioni ma semplici definizioni di comportamento, il che lascia liberi i progettisti di definire le API come meglio credono. Sono comuni nei sistemi UNIX.

Nel sistema Windows 2000 viene impiegato il modello uno a uno e ciascun thread contiene un identificatore, un insieme di registri, una pila d'utente e una del sistema e un'area di memoria privata.

Nel sistema Linux viene usata il termine 'task' per processi e thread e si usa la chiamata del sistema <clone>, che crea un processo distinto che condivide lo spazio d'indirizzi del processo chiamante.

Nel linguaggio Java i thread sono gestiti dalla JVM e possono essere creati in due modi: creando una nuova classe derivata dalla classe Thread o sovrascrivendo il metodo run di quella classe.

Lezione 7 (Cap.5)

L'accesso concorrente a dati condivisi può causare incoerenza negli stessi e per prevenire ciò bisogna elaborare i giusti meccanismi preventivi: nella soluzione del problema di produttori e consumatori con memoria limitata si può pensare di aggiungere una variabile intera "counter" inizalizzata a 0 che va incrementata o decrementata in modo atomico, cioè senza interruzione.

Poiché si finirebbe comunque in una situazione di 'race condition', in cui i risultati dipendono dall'ordine degli accessi, c'è bisogno di sincronizzare i processi concorrenti.

Considerando n processi, sappiamo che ognuno di essi ha una

sezione critica in cui può modificare variabili comuni, per cui deve essercene uno solo in tale fase tra tutti i processi in esecuzione in quel momento. Si adottano diversi accorgimenti:

- Mutua Esclusione: se un processo è nella sua fase critica, allora esso è l'unico in tale fase.
- Progresso: se nessun processo è in sezione critica e qualcuno vuole entrarvi, allora sono gli altri processi fuori da sezione critica a scegliere chi può entrare per primo.
- Attesa Limitata: un processo che ha già richiesto l'ingresso può essere 'scavalcato' non troppe volte volte.

Nell'algoritmo del fornaio viene assegnato un numero a ciascun processo prima di entrare nella fase critica e viene data la precedenza a chi ha il numero più basso; a parità di numero viene servito per primo il processo che lo ha ricevuto per primo.

Un semaforo S è una variabile intera che viene usata per eseguire una sincronizzazione che non richiede attesa attiva. Vi si accede tramite due operazioni atomiche predefinite, <wait> e <signal>.
Si verifica una situazione di stallo (deadlock) quando due processi attendono indefinitamente un evento che può essere causato solo da uno dei processi in attesa, mentre si verifica un'attesa indefinita (starvation) quando si attende nella coda di un semaforo indefinitamente.
Il semaforo può sia essere implementato come contatore ed avere un valore appartenente ad un dominio logicamente non limitato oppure come binario ed avere come valori possibili solo 0 e 1.

Lezione 8 (Cap.8)

Il problema dello stallo avviene quando si ha un insieme di processi bloccati, con tutti che detengono una risorsa e ne attendono un altra possseduta da un altro processo.
Si può considerare l'attraversamento di una sezione di un ponte che consente il passaggio in una sola direzione: ciascun lato è una risorsa e se si verifica uno stallo, cioè si cerca di attraversare il ponte da ambo i lati contemporaneamente, una delle macchine (o più) effettua una retromarcia, cioè avviene un 'rollback' che porta a un ristabilirsi di uno stato sicuro.

Affinché ci sia uno stallo c'è bisogno che si verifichino 4 condizioni contemporaneamente:

- Mutua esclusione: solo un processo per volta può usare una risorsa.
- Possesso e attesa: un processo in possesso di almeno una risorsa ne attende un'altra già in possesso di un altro processo.
- Impossibilità di prelazione: una risorsa può essere rilasciata solo al termine del processo che la possiede.
- Attesa circolare: esiste un insieme di processi in cui ognuno attende una risorsa dal successivo e l'ultimo processo Pn la attende dal primo processo P0.

Per disegnare un grafo di assegnazione delle risorse bisogna considerare due insiemi: l'insieme V dei vertici, suddiviso a sua volta nel sottoinsieme P dei processi del sistema ed il sottoinsieme R dei tipi di risorse del sistema, e l'insieme E degli archi.
Un arco orientato che va da Pi a Rj è un arco di richiesta, viceversa da Rj a Pi è di assegnazione.
Se il grafo non contiene cicli allora non si verificano stalli, se invece sono presenti dei cicli: se c'è una sola istanza per tipo di risorsa si verifica uno stallo, mentre se vi sono più istanze per tipo di risorsa si potrebbero verificare degli stalli.

Per la gestione degli stalli ci sono diverse soluzioni, ad esempio assicurarsi che il sistema non vi entri mai, oppure consentire la presenza di stalli, ma individuandoli ed eseguendo il ripristino, o ancora ignorare il problema 'fingendo' che gli stalli non possano mai verificarsi (più comune).

Per prevenire gli stalli ci si deve assicurare che almeno una delle 4 condizioni non possa verificarsi:

- Mutua esclusione: deve valere solo per le risorse non condivisibili, poiché non è richiesta nel caso di quelle condivisibili.
- Possesso e attesa: ogni processo, prima di iniziare, deve richiedere tutte le risorse che gli servono e devono essergli assegnate oppure viene concesso ad un processo di richiedere risorse solo se non ne ha già; comporta un basso utilizzo delle risorse e c'è rischio di attesa indefinita.
- Impossibilità di prelazione: se un processo richiede risorse che non gli si possono assegnare viene effettuata la prelazione su tutte le risorse in suo possesso, che vengono aggiunte alla lista delle risorse che il processo attende, e può essere riavviato solo quando saranno tutte ottenibili.
- Attesa circolare: viene imposto un ordinamento totale all'insieme di tutti i tipi di risorse e un ordine crescente di numerazione per le risorse richieste da ciascun processo.

Un metodo alternativo potrebbe essere quello di richiedere ulteriori informazioni sui modi di richiesta delle risorse: il più semplice ed utile impone ai processi di dichiarare il numero massimo delle risorse di ciascun tipo di cui necessita.
L'algoritmo per evitare lo stallo controlla dinamicamente lo stato di assegnazione delle risorse, cioé quante sono disponibili e quante assegnate, per garantire che non possa esistere una condizione di attesa circolare.

Uno stato si dice sicuro quando il sistema può assegnare risorse a ciascun processo (fino al suo massimo) in un certo ordine e impedire il verificarsi di uno stallo. Questo avviene quando esiste una sequenza sicura: se per ogni Pi appartenente alla sequenza dei processi, le richieste di Pi si possono soddisfare impiegando le risorse attualmente disponibili + quelle possedute dai processi Pj con $j < i$.

Nel caso le risorse non siano subito disponibili, Pi può attendere che i processi precedenti terminino.
Riprendendo lo schema con i grafi, possiamo indicare l'arco da P1 a Rj con una linea tratteggiata se la risorsa può essere richiesta e chiamarlo arco di reclamo. Quando la risorsa è richiesta si passa ad un arco di richiesta e quando il processo la rilascia si passa da arco di assegnazione ad arco di reclamo.
Le risorse devono essere richiamate a priori nel sistema.

Nell'algoritmo del banchiere ogni processo deve dichiarare a priori il numero massimo delle istanze di ciascun tipo di risorsa necessaria, può dover attendere quando le richiede ed ha un intervallo di tempo definito entro cui restituirle.

Dato n numero di processi del sistema ed m il numero dei tipi di risorsa, abbiamo le seguenti strutture dati:

- Disponibili, un vettore di taglia m che indica il numero delle istanze disponibili per ciascun tipo di risorsa.
- Massimo, matrice n*m che definisce la richiesta massima di ciascun processo.
- Assegnate, matrice n*m che definisce il numero delle istanze di ciascun tipo di risorse attualmente assegnate.
- Necessità, matrice n*m che indica la necessità residua di risorse relativa ad ogni processo.

Un sistema che non si avvale di algoritmi di stallo deve prevederne uno per esaminare lo stato del sistema ed un altro che lo ripristini dalla condizione di stallo.

Nel caso di istanza singola di ciascun tipo di risorsa si può implementare un grafo d'attesa in cui i nodi sono processi e Pi -> Pj indica che Pi è in attesa di Pj. L'algoritmo controlla periodicamente il grafo alla ricerca di cicli, richiedendo n^2 operazioni, con n numero dei vertici.
Se invece ci sono più istanze si possono usare un vettore Disponibili di lunghezza m che indica il numero delle istanze disponibili per ciascun tipo di risorsa e due matrici n*m Assegnate e Richieste, che indicano rispettivamente il numero delle istanze di ciascun tipo di risorse correntemente assegnate e la richiesta attuale di ciascun processo.

Per sapere quando è necessario ricorrere all'algoritmo di rilevamento bisogna considerare la frequenza con cui si presume possano avvenire gli stalli ed il numero dei processi che sarebbero influenzati dallo stallo. Non è conveniente richiederlo in momenti arbitrali, poiché potrebbero esserci molti cicli e, di solito, non si può dire quale dei tanti sia il vero 'responsabile'.
Un altra strada sarebbe quella di terminare tutti i processi in stallo, uno per volta, secondo alcuni criteri: priorità dei processi, tempo trascorso dalla computazione e tempo ancora necessario, quantità e tipo di risorse impiegate, quantità di risorse ancora necessarie, numero di processi da terminare.

Si può anche effettuare una prelazione di risorse: viene scelta una 'vittima' cercando di minimizzare i costi per effettuare un rollback, ma si rischia di scegliere sempre lo stesso processo, portando ad una attesa indefinita.

Combinare i tre approcci di base consente l'uso dell'approccio ottimale per ciascuna delle risorse del sistema; in più si possono dividere le risorse in classi ordinate gerarchicamente ed applicare la tecnica più appropriata in ciascuna classe.