02 Maggio 2023

ARCHITETTURE DI CALCOLO LEZIONE 13

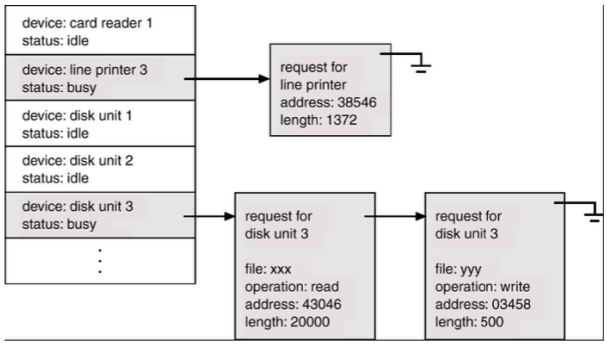
Sistemi I/O, protezione della CPU, strutture dati kernel

TABELLA DI STATO DEI DISPOSITIVI

Nei sistemi operativi, soprattutto in quelli di tipo time-sharing, è spesso necessario il passaggio da un processo all'altro; per questo motivo, l'hardware della CPU dispone di una o più linee di richiesta di interruzione. Il gestore delle interruzioni ne rileva la natura, salva le informazioni di stato, invoca la routine di servizio opportuna e, infine, esegue un'istruzione di ritorno e ripristina lo stato della CPU. Ricordiamo che i processi sono quantizzati nel tempo, ossia hanno un tempo limitato oltre il quale il processo stesso viene killato.
Nei sistemi di ultima generazione troviamo un hardware dedicato al controllo delle interruzioni. Ciò porta i seguenti vantaggi:

- Efficienza nel recuperare la routine di servizio;
- Possibilità di posticipo della gestione degli interrupt durante un'elaborazione critica;
- Interruzioni mascherabili, cioè temporaneamente disattivabili (inviate da controllori), e non mascherabili (condizioni di errore);
- Interruzioni multilivello, con diverse priorità, per reagire con l'urgenza appropriata.

In tutti i sistemi operativi è presente la tabella di stato, ossia una struttura dati con il compito di indicare l'elenco dei dispositivi in uso e per ognuno la lista di richieste concatenate da eseguire; il tutto è fatto al fine di eseguire in processi in modo ordinato. Ad esempio, vediamo nella tabella a sinistra il disco unità 3 che risulta occupato e dalla freccia sono indicate le varie richieste che sta svolgendo.
Un esempio tipico è la coda di stampa di una stampante che prevede la formulazione di una lista ordinata di azioni di stampa.



STRUTTURA I/O

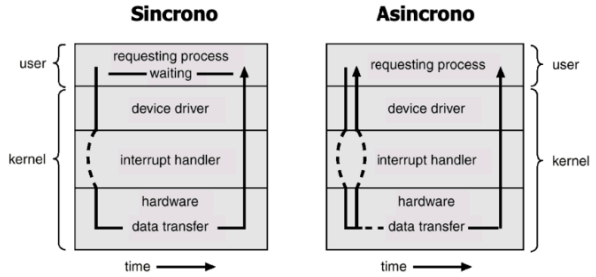
Una percentuale cospicua del codice del SO è dedicata alla gestione dell'I/O poiché:

- L'interazione con i dispositivi di I/O è fondamentale per il progetto di un sistema affidabile ed efficiente;
- Vi è una grande variabilità nei dispositivi di ingresso/ uscita che vanno gestiti.

Ciascun controllore si occupa di un particolare tipo di dispositivo (può gestire una o più unità adesso connesse). Fondamentale è il driver del dispositivo, che è il "sistema operativo" del controllore. Il sistema operativo non è in grado di modificare il driver del dispositivo un quanto non ne conosce il codice.
Inseguito ad una richiesta di I/O, da parte di un processo utente, si verifica un'interruzione. Dopo l'inizio dell'operazione di I/O, il flusso di esecuzione può seguire due percorsi distinti:
I/O sincrono: nel caso più semplice, si restituisce il controllo al processo utente solo dopo il completamento dell'operazione di I/O. La CPU è mantenuta in stato idle (inutilizzata) da una wait; ad ogni istante, si può avere una sola richiesta di I/O in esecuzione. In questa fase ad un comando di I/O segue una fase di wait della CPU, con perdita di tempo (modello dei "vecchi" sistemi blatch).

I/O asincrono: in alternativa, si può prevedere la restituzione immediata del controllo al (ad altro) processo utente: in questo modo l'I/O può proseguire mentre il sistema esegue altre operazioni. Questa è la modalità usata oggi che permette, ad esempio, di stampare in background mentre si aprono altri programmi.

A sinistra vediamo a confronto i due tipi di flussi:



1.Nel modello sincrono vediamo che se in un dato momento si ha la richiesta di un processo di stampa:
•Tale richiesta passa tramite una system call al kernel del sistema operativo;
•Il sistema operativo fa uso del device driver per avviare l'azione di stampa;
•Dopodiché la stampa inizia, mentre il processo utente rimane in waiting;
•A stampa completata il processo utente riprende il controllo.

2. Nel modello asincrono abbiamo invece:
- La richiesta passa tramite una system call al kernel;
- Il controllo ritorna immediatamente al processo utente e, contemporaneamente, ha inizio la stampa;
- Così, mentre il processo utente usa la CPU per eseguire altre richieste, l'hardaware esegue la richiesta di stampa.

È preferibile il modello asincrono idealmente, ma nella pratica non sempre è utilizzabile. Ad esempio, se ho necessità di eseguire dei calcoli su un file da leggere posso compiere i calcoli solo a lettura completata. Qui entrano in gioco i programmi multi-tread moderni con i tread (componenti) che lavorano in parallelo.
Per poter gestire la modalità asincrona è necessario un buffer, ossia una memoria interna sempre sincronizzata che permette di mantenere in stand by le operazioni da eseguire (secondo una logica fifo) fin quando non arriva "il suo turno" di esecuzione. Un controllore di dispositivo dispone di una propria memoria interna (buffer), e di un insieme di registri specializzati. Il controllore è responsabile del trasferimento dei dati tra i dispositivi periferici a esso connessi e la propria memoria interna. I sistemi operativi in genere possiedono, per

ogni controllore di dispositivo, un driver del dispositivo che si coordina con il controllore e funge da interfaccia uniforme con il resto del sistema.

Struttura I/O

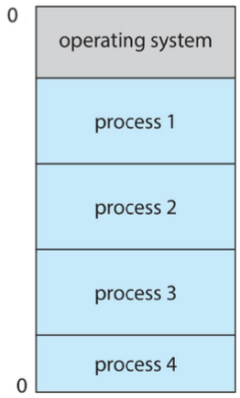
-Per avviare un'operazione di I/O, il driver del dispositivo carica gli appropriati registri all'interno del controllore, il quale, dal canto suo, esamina i contenuti di questi registri per determinare l'azione da intraprendere (per esempio "leggi un carattere dalla tastiera").
- Il controllore comincia a trasferire i dati dal dispositivo al proprio buffer. A trasferimento completato, il controllore informa il driver, tramite un'interruzione, di avere terminato l'operazione. -Il driver passa quindi il controllo al sistema operativo, restituendo i dati.
Questa forma di I/O guidato dalle interruzioni è adatto al trasferimento di piccole quantità di dati, ma in caso di trasferimenti massicci può generare un pesante sovraccarico (overhead); si pensi, per esempio, all'I/O da e verso il disco.
Soluzione: Accesso diretto alla memoria (DMA)
-Il controllore trasferisce un intero blocco di dati dal proprio buffer direttamente nella memoria centrale, o viceversa, senza alcun intervento da parte della CPU. Infatti, la CPU verrebbe sempre richiamata in quanto prevede un componente detto parte di controllo che guida l'accesso al bus. Tramite la DMA, la CPU non interviene e non termina i suoi cicli.

CPU-scheduling

In generale nei sistemi operativi abbiamo diversi tipi di scheduling (della CPU,del disco, della memoria, a breve termine,a medio termine, ecc.). Il principale tra gli scheduler è il CPU-scheduler o scheduler a breve termine, che ha il compito di gestire le operazioni successive da eseguire una volta che la CPU è libera.
La scelta degli scheduler è olistica: la preferenza di ottimizzazione di uno scheduler rispetto ad un altro influisce negativamente sulla produttività degli altri. Ad esempio, se si mette la modalità di risparmio energetico (scheduler per il risparmio dell'energia) si ha un aumento del tempo di risposta agli input (diminuzione delle prestazioni).
Nei sistemi time-sharing, in particolare, in cui il tempo d'uso della CPU viene diviso in più task (da qui il termine multitasking), il tempo di risposta accettabile è al di sotto di 1 secondo. La CPU-scheduling serve a ridurre tale tempo di risposta.
Nota: la memoria virtuale fa credere alla macchina che ha più memoria di quella realmente posseduta ma una macchina a memoria fisica maggiore è più prestante per il tempo risparmiato nel continuo scambio(swap) di dati dalla memoria al disco e viceversa.

Configurazione della memoria di un sistema multi- programmato

All'interno di un processo abbiamo 2 componenti:



•I dati
•Il codice del programma da eseguire
Se per qualche ragione il ciclo while del primo processo va al di fuori della memoria che ha a disposizione, esso va ad intaccare la memoria degli altri processi. Per ovviare a tale problema si sono introdotti degli appositi controllori che in questi eventuali casi killa il precesso fuori posto.
Importante è che mentre il sistema operativo (sistema kernel) può usufruire della memoria di tutti i processi, i processi hanno a disposizione solo il loro spazio di memoria. Per questo motivo il codice del kernel deve essere scrupolosamente controllato.

Sistemi Real-Time

Sistemi (in tempo reale) con vincoli temporali ben definiti sull'elaborazione e sull'accesso alle risorse. Sono spesso usati per controllare dispositivi in applicazioni dedicate come:
- Gestione di macchine o di robot,
- Gestione di immagini in medicina,
- Sistemi di controllo militare,
- Gestione di dati scientifici,
- Algoritmi di scheduling specifici.

Sistemi portatili-mobili

Comprendono tutti i dispositivi dotati di batteria mobile, che li rende portatili:
- Personal Digital Assistants(PDAs)
- E-books
- Smartphone
- Sensori
- RFID

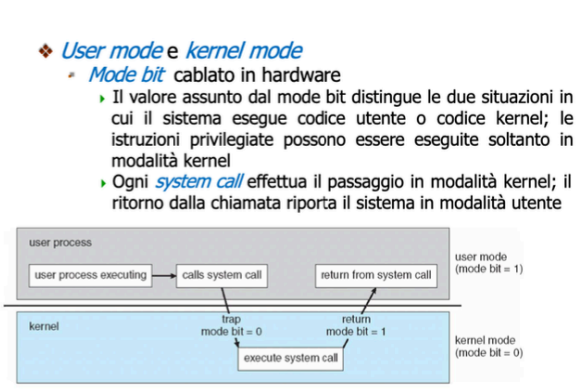
Problemi: Memoria limitata, Processori lenti, Display piccoli
Elisa: chatbot degli anni 80' che girava su common64

PROTEZIONE HARDWARE

Un sistema operativo deve impedire che il cattivo funzionamento di un programma influenzi la corretta esecuzione del sistema operativo stesso e di altri programmi. L'insieme delle tecniche attuate a tale scopo prende il nome di protezione hardware.
Approcci alla protezione:
- Funzionamento Dual-Mode,
- Protezione dell'I/O,
- Protezione della Memoria,
- Protezione della CPU.

Dual-mode

Il sistema operativo è interrupt-driven e può ricevere, oltre agli interrupt hardware provenienti dai device, interrupt di errori nel software o richieste di servizi che creano situazioni di eccezione o trap (divisioni per zero, processi in loop, processi che tentano di modificare lo spazio di memoria dedicato ad altri processi o al sistema operativo).
La modalità operativa dual-mode permette al SO di proteggersi e di proteggere le varie componenti del sistema di calcolo.
Nel sistema operativo esistono due modalità:



•**Modo Utente**: esecuzione svolta per conto dell'utente; solo alcune attività sono permesse esclusivamente nella RAM.
•**Modo kernel o system o monitor**: esecuzione svolta per conto del sistema operativo.
Nella modalità utente non si possono eseguire istruzioni che possano creare malfunzionamenti.
Per switchare dalla modalità monitor a quella utente (ed evitare errori di sistema) si è introdotto il mode bit cablato in hardware.

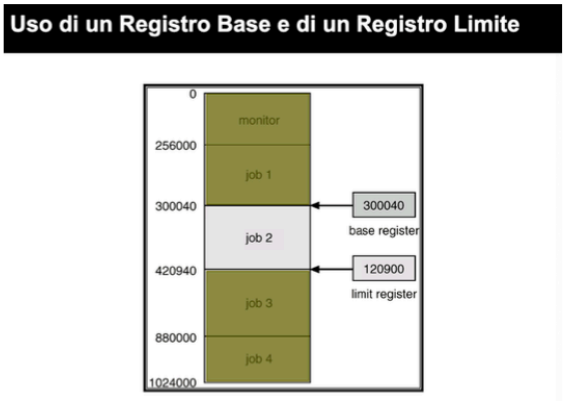
Protezione dell'I/O

Tutte le istruzioni di I/O sono istruzioni privilegiate e bisogna assicurare che un programma utente non possa ottenere il controllo del computer in modo monitor. Questo potrebbe succedere se le operazioni di I/O venissero eseguite in modo utente; le system call si usano a questo scopo.

Protezione della Memoria

Occorre proteggere aree di memoria critiche (come il vettore delle interruzioni e le routine di servizio degli interrupt) al fine di salvaguardare la macchina. Per fare ciò, si usano due registri per determinare l'intervallo di valori corretto che un programma può accedere:

- **Registro Base**: contiene l'indirizzo iniziale della memoria che un programma può accedere;
- **Registro Limite**: contiene la dimensione dell'intervallo.



Ad esempio, il job 2 ha una memoria che va dalla posizione 300040 alla 420940; il registro base sarà l'indirizzo iniziale, ossia 300040, mentre il registro limite sarà la lunghezza dell'intervallo, cioè 120900. Questi due registri vengono impostati e mantenuti dal sistema operativo, e dunque non possono essere modificati dal processo (che quindi non può accedere alla memoria del processo adiacente senza l'intervento del sistema operativo).

Il sistema operativo, ogni volta che viene operata una richiesta di accesso alla memoria da un job, deve eseguire un controllorappresentato dal grafico a sinistra. Quando la CPU genera un indirizzo, innanzitutto si deve verificare che esso sia maggiore o uguale dell'indirizzo di base:
•Se è falso, si genera un errore ed il processo viene ucciso;
•Se è vero, si esegue il controllo che l'indirizzo sia minore del registro base + il registro limite:
1.Se è falso il job viene killato;
2.Se è vero il processo accede alla memoria.

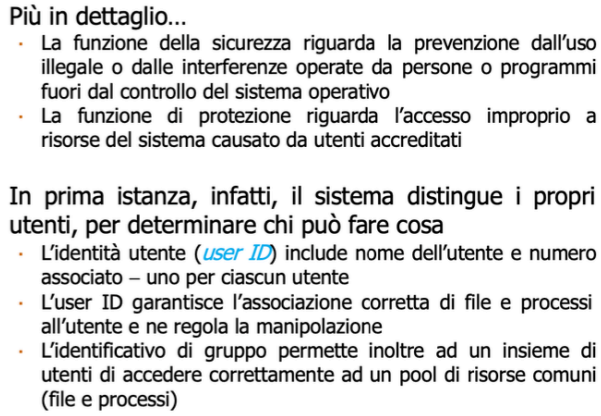
Da dove genera gli indirizzi la CPU? Da una cella della RAM (o direttamente o come calcolo di successivi). Ricordiamo che la CPU genera continuamente indirizzi e dati.
Questa protezione della memoria ha un costo temporale non indifferente. I sistemi operativi in genere prevedono dei sistemi di protezione della memoria ma nulla vieta di creare dei sistemi operativi con vincoli di protezione meno rigidi a favore della velocità di risposta.
Ogni volta che la CPU genera un indirizzo di processo, le istruzioni di caricamento e modifica dei registri base e limite vengono eseguite dal sistema operativo in modalità monitor (non dal modo utente!!!).

Protezione della CPU

È fondamentale che un processo utente non usi indefinitamente la CPU. La soluzione è l'uso di un timer decrescente che permette l'uso della CPU da parte di un processo in modo limitato. La scelta del timer è compito del sistema operativo.

PROTEZIONE E SICUREZZA

Se diversi utenti usufruiscono dello stesso elaboratore che consente l'esecuzione simultanea e concorrente dei processi, l'accesso alle risorse del sistema di calcolo dovrà essere disciplinato da regole imposte dal SO.
File, segmenti di memoria, CPU, altre risorse possono essere manipolate solo dai processi che hanno ottenuto apposita autorizzazione dal SO.
La protezione nel sistema operativo avviene si realizza mediante due concetti:



•**Politiche**: definiscono gli obiettivi dei controlli (es. un file dell'utente X può essere aperto solo dall'utente X);
•**Meccanismi**: gli strumenti per la loro applicazione.
Due macchine possono avere la stessa politica ma meccanismi diversi.
In un sistema operativo siamo soggetti, soprattutto nel mondo delle reti, a molti rischi. Prima di internet i virus erano trasmessi tramite installazione di programmi corrotti, oggi sulla rete abbiamo anche gli attacchi:

- **DOS (Denial-of-Service)**: malfunzionamento dovuto ad un attacco informatico in cui si esauriscono deliberatamente le risorse di un sistema di calcolo che fornisce un servizio, fino a renderlo non più in grado di erogarlo (es. quando si cercano di comprare dei biglietti di un concerto). Una modalità di attacco è l'infezione di tante macchine a creare una bot-net (rete di macchine zombie) che in un dato giorno ad una precisa ora mandano in gruppo delle semplici richieste ad un server mandandolo in sovraccarico. Una modalità di protezione è l'apertura di una finestra blank prima di accedere al sito che, facendo perdere del tempo alla macchina, distingue l'utente reale dal bot.
- **Trojan**: programmi che hanno una funzione conosciuta legittima e una funzione dannosa nascosta (ad esempio attivare un attacco DOS).
- **Worm**: malware in grado di autoreplicarsi (può essere un trojan) su altri dispositivi.
- **Virus**: porzioni di codice dannoso che si legano ad altri programmi del sistema per diffondersi.
- **Ransomware** (ransom [riscatto] + software): criptano i file e chiedano un riscatto con timer in bitcoin. È fondamentale per salvaguardare i dati eseguire dei backup offline.

STRUTTURE DATI DEL KERNEL

La struttura dati più usata nel SO è l'array. Un array è una semplice struttura dati in cui ogni elemento è direttamente accessibile tramite un indice.

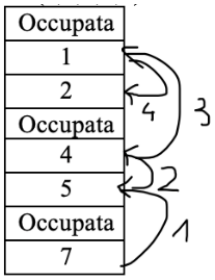
- La memoria principale è costruita come un array, così come il vettore degli interrupt.

L'alternativa agli array possono essere le liste concatenate.
L'uso di uno dei due formati è dettato dalla diversa necessità del programma.
Voglio andare nella cella n di una lista quanto tempo impiego ?
O(n), cioè di un tempo di ordine n.
L'array permette, invece, di andare direttamente in una cella precisa senza passare per le celle precedenti (costo=1).
Esempio:
Dato un vettore V di lunghezza n, per metterlo in memoria è necessario avere n celle libere consecutive in modo da poterlo memorizzare:

Occupata
7
5
4
1
2
Occupata

V=(7,5,4,1,2)

Ovviamente nel caso di un vettore di grandi dimensioni non è detto trovare n celle adiacenti libere. V[3]=5 costo=1
Se prendiamo, invece, una lista da memorizzare, l'ordine della lista non è mantenuto tramite l'adiacenza delle celle ma tramite dei collegamenti tra celle:



L=(7,5,4,1,2)

Le liste concatenate possono contenere dati di diversa natura e dimensione (collezionati all'interno di record)
Potenziale svantaggio: il costo della ricerca è $O(n)$
Le liste sono talvolta utilizzate direttamente dagli algoritmi del kernel, ad esempio nell'allocazione concatenata di file o nell'algoritmo Clock per la sostituzione delle pagine
Più frequentemente le si utilizza per implementare pile e code

a

I vantaggi della lista sono:
•Possibilità di memorizzare dati quando la memoria è frammentata (non ci sono celle libere di fila).
•Eseguire operazioni di rimozione/modifica/aggiunta di dati più facilmente.

Svantaggi:
•Lentezza
•Costo di ricerca maggiore

Lo stack viene acceduto con politica LIFO (Last In First Out) e le operazioni di inserimento ed estrazione (cancellazione) vengono rispettivamente dette *push* e *pop*

- Il SO utilizza lo stack per gestire le chiamate di funzione
  - Al momento della chiamata, si inseriscono nello stack l'indirizzo di ritorno, le variabili locali ed i parametri attuali
  - ...che vengono recuperati dalla funzione
- Anche il cambio di contesto viene gestito tramite stack

Le code si accedono in modalità FIFO (First In First Out)

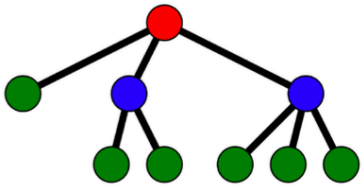
- Il SO gestisce come una coda, per esempio, i documenti inviati ad una stampante o i processi in attesa di ottenere l'accesso alla CPU

b

Altre strutture sono la pila (stack) e la coda. Le slide riportate (a,b) non vengono trattate.

Gli alberi sono utilizzati per organizzare i dati in maniera gerarchica e sono basati sulla relazione causale padre–figlio

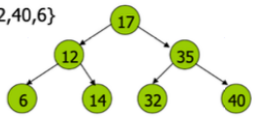
Gli alberi possono avere diversa *arietà*, cioè, da ciascun nodo, possono dipartirsi un numero diverso di archi



Un'altra importante struttura dati è l'albero, utilizzato per organizzare i dati in maniera gerarchica e sono basati sulla causale padre-figlio.
L'albero è interessante perché permette di stabilire i diritti dei processi che si trasmettono dal processo padre ai processi figli; tale meccanismo permette di creare processi che hanno diritti limitati a quelli posseduti dal processo padre.
Più avanti studieremo come avviene la generazione dei figli tramite il meccanismo di fork (biforcazione).
Se ogni padre genera al più due figli si parla di albero binario. Dato un padre, il figlio a sinistra ha valore minore del padre mentre il figlio a destra ha valore maggiore del padre.

- Sono alberi binari di ricerca quelli per cui vale la relazione d'ordine $figlio_{sinistro} < padre < figlio_{destro}$
  - Nel caso peggiore la ricerca costa $O(n)$, ma, a patto di mantenere l'albero "bilanciato" il costo medio di questa operazione scende a $O(\log_2 n)$

{17,35,12,14,32,40,6}



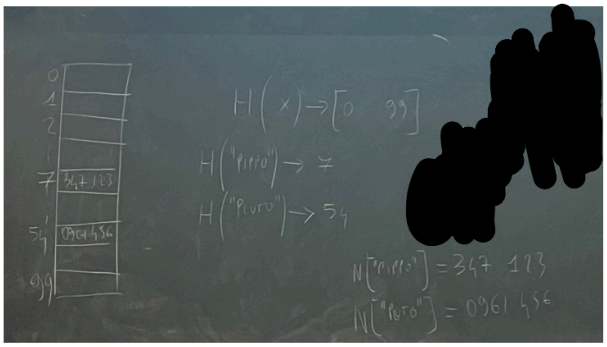
- Linux utilizza un albero binario di ricerca bilanciato nel suo algoritmo di scheduling della CPU

Un albero binario permette di eseguire una ricerca con costo O(log2n).
Tale albero permette di eseguire ricerche in modo molto rapido e per questo viene anche detto albero di ricerca.

Funzione di Hash

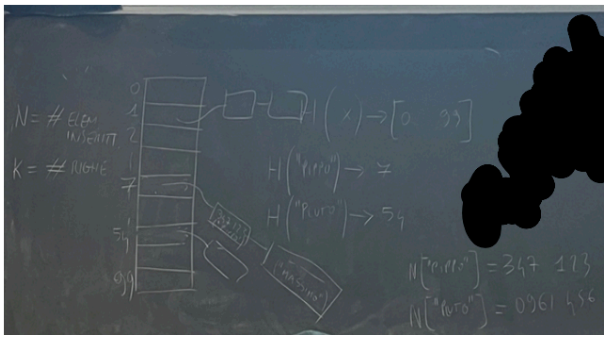
La funzione di hash riceve in input un valore e ne dà in output un altro. Generalmente il valore in output è un valore minore di quello ricevuto in input. Un'applicazione è la tabella di hash. Ad esempio: si riceve in input un valore compreso tra 1 e 1000 (anche un nome) ed in

output 7; la cosa importante è che ogni qualvolta in input si richiama tale valore l'output sia sempre 7.



Prendiamo ad esempio una tabella di Hash che va da 0 a 99 (100 celle) che memorizza numeri di telefono. Presi i numeri di "Pippo" e "Pluto", tramite la funzione di hash memorizziamo i numeri di telefono nelle celle 7 e 54; quando richiameremo la funzione H("Pippo") verrà così richiamata la cella 7 e verrà stampato il numero di telefono corrispondente.

La caratteristica principale della funzione di hash è quella di permettere individuare in tempo costante la cella contenente l'informazione ricercata.
Cosa succede se la funzione di hash calcola lo stesso valore di hash per diversi valori di input?

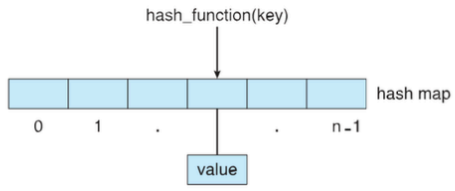


L'input è di lunghezza qualsiasi, l'output è da 0 a 99; possono esserci dei conflitti. Per gestire le collisioni si modifica la struttura dati:
•Nella cella della tabella di hash non inseriamo direttamente l'input (nell'esempio il numero di telefono), bensì una lista concatenata di dati detta lista di collisione.
•Nell'esempio precedente, nella cella 7 troviamo una lista di numeri da richiamare.

Ipotesi: K sottoinsieme dei numeri naturali
Possibile funzione di accesso:

$$h(k) = k \text{ MOD } n,\ k \in K$$

- Valore della funzione sempre compreso fra 0 e $n-1$



Il costo della ricerca così non è più O(1), ma diventa O(N/K), dove N è il numero di elementi della lista e K il numero di righe della tabella di Hash.
Per ottimizzare la ricerca bisognerebbe aumentare K, cioè la lunghezza della tabella si Hash, ma ciò non può essere portato all'infinito perché la sua realizzazione fisica lo impedisce.

Problema – Memorizzare in maniera opportuna un insieme di dati, tipicamente sotto forma di record, in modo da poter reperire un qualsiasi elemento dell'insieme con un numero "piccolo" di tentativi

- Cosa significa "piccolo" ?
  - Indipendente (o quasi) dalla dimensione della tabella su cui si effettua la ricerca, quindi con una complessità in tempo pari a $O(1)$

Funzioni hash

- $h: K \rightarrow \{0, 1, 2, \ldots, n-1\}$ (*to hash*, in inglese, significa sminuzzare)
  - K: insieme dei valori distinti che possono essere assunti dalle chiavi dei record
  - n: dimensione del vettore in cui si intende memorizzare la tabella

Nella prossima lezione verrà spiegata l'applicazione di tale funzione.