Università degli Studi di Cagliari
Corso di Laurea in Ingegneria Elettronica

ELEMENTI DI INFORMATICA

http://agilegroup.eu

A.A. 2015/2016

Docente: **Michele Marchesi**

DATI PERMANENTI E MEMORIE DI MASSA

Sommario

- **Dispositivi fisici di memoria di massa:**
  - **nastri**
  - **dischi**
  - **memorie flash**
- **I file**
- **Il file system**
- **Organizzazione interna dei file**

Memorie permanenti o *di massa*

- **Memoria di massa o persistente: memorizza dati che sono mantenuti anche in caso di spegnimento del calcolatore**
- **Dispositivi che:**
  - **memorizzano grandi quantità di dati**
  - **forniscono un accesso efficiente**
- **Tipologie delle memorie di massa:**
  - **memorie ad accesso sequenziale: nastri**
  - **memorie ad accesso diretto: dischi e tamburi**
  - **memorie a stato solido: penne USB, ecc; usano lo stesso approccio di memorizzazione dei dischi**

Nastri magnetici

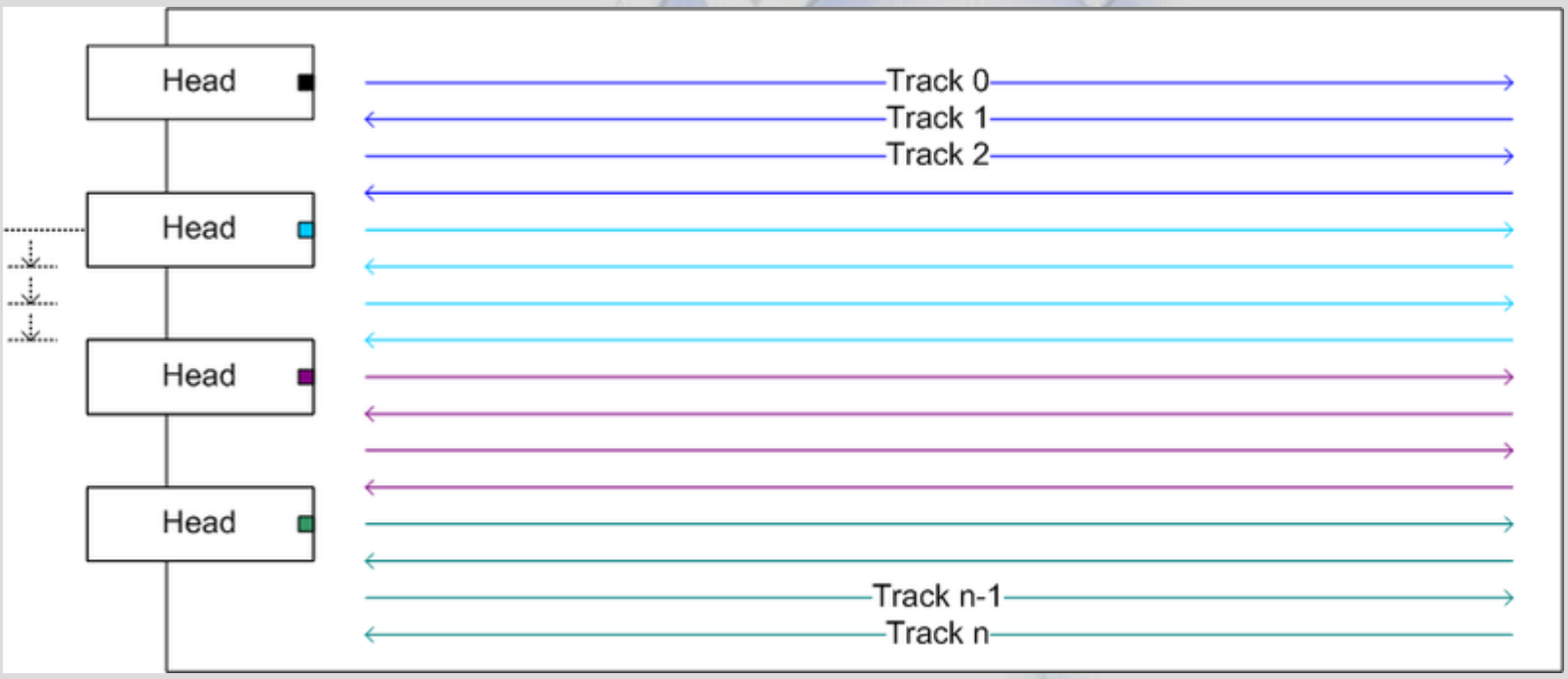
- **Accesso *sequenziale* ai dati**
- **Memorizzabili grandi masse di dati**
- **Molto veloci per leggere/ scrivere dati consecutivi**
- **Lentissimi per accedere a dati disposti in modo arbitrario (accesso *random*)**
- **Ormai poco usati, soprattutto per *backup***
- **I nastri moderni sono in cartuccie (da qualche Tbyte)**

Nastri magnetici (2)

- **I nastri registrano i dati su materiale magnetico (Fe_2O_3 e CrO_2), posto su strato di mylar (PET)**
- **I dati sono scritti e letti da *testine magnetiche* che sfiorano il nastro ma non lo toccano**
- **I primi nastri avevano testine fisse e tracce poste lungo la direzione del nastro:**
  - **6 o 8 tracce con i dati, una per il *bit di parità***
  - **il bit di parità rileva errori di scrittura/lettura**
- **Indicatori di BOT (Beginning of Tape) e EOT (End of Tape)**
- **Dati raggruppati in *blocchi*, separati da *gap*; un blocco viene letto e scritto in un'unica operazione**

Tracce a serpentina

- **I nastri moderni possono avere anche oltre 350 tracce, con accesso a *serpentina:***
  - **le testine sono meno delle tracce**
  - **arrivati in fondo al nastro, le testine si spostano e percorrono un'altra traccia, in senso opposto, e così via**



Parametri dei nastri magnetici

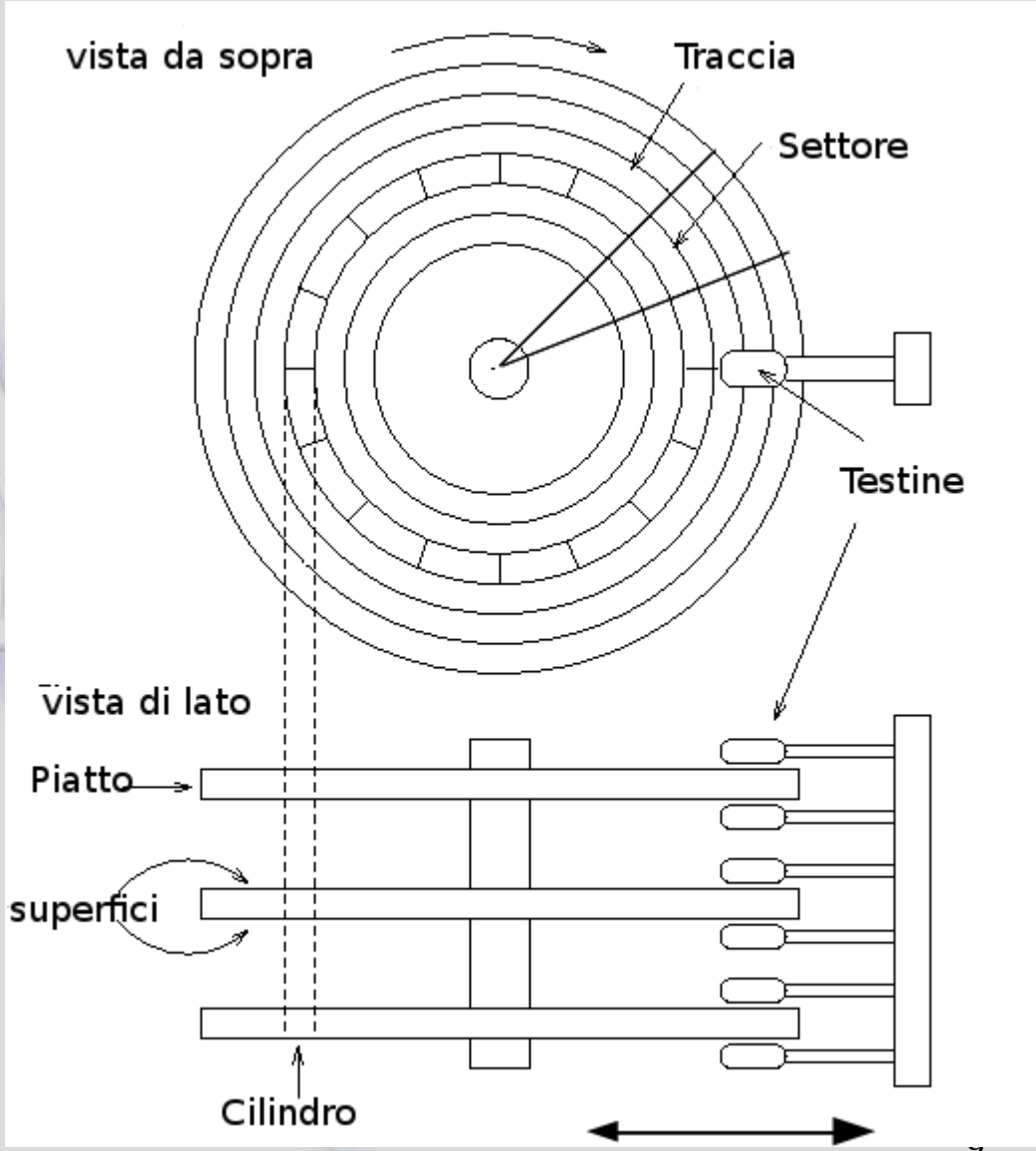
- **Densità di memorizzazione in bit per inch (bpi): ad es. 1024 bpi nei primi nastri, anche 100 Mbpi ora**
- **Capacità di memorizzazione del nastro in byte (densità in bpi x lunghezza / 8): es. 4 TB**
- **Velocità di movimento in lettura e scrittura (inch/sec), ad es. 100 in/s**
- **Data rate: byte/s, ad es. 90 KB/s (IBM 767, 1964), 250 MB/sec (IBM 3592, 2008)**
- **Lunghezza del nastro (es. 100-1000 m.)**
- **Dimensione del blocco**
- **Compressione dei dati o no**

Dischi magnetici

- **I dischi magnetici memorizzano l'informazione su un disco (o un *disk-pack*) che ruota**
- **Disk-pack: un certo numero di dischi coassiali, ciascuno con due facce magnetizzate**
- **Ogni faccia ha un insieme di *tracce* concentriche, divise in *settori* corrispondenti (che coprono uno "spicchio" del disco**
- **Delle testine magnetiche, una per faccia, si muovono in senso radiale, tutte insieme**
- **Accesso diretto: ogni settore può essere raggiunto in pochi millisecondi dalla testina**

Dischi magnetici

- **L'inizio e la fine dei settori sono marcati con sequenze di bit (formattazione)**
- **I dati sono letti e scritti in *blocchi* con un'unica operazione**
- **Una zona di memoria (buffer) contiene copia dei settori in lettura/scrittura per velocizzare le operazioni**



Tempi di accesso

- **Per accedere a un settore:**
  - **le testine si spostano sulla traccia (*seek*)**
  - **si aspetta che il settore, ruotando, passi sotto la testina (*latenza*): in media ½ del *t* di rotazione**
  - **i dati sono letti e trasferiti elettronicamente, o trasferiti e scritti (*trasferimento*)**
- **Tempo di accesso:** $t_{seek} + t_{latenza} + t_{trasf}$
- **Leggendo più settori consecutivi, seek e latenza sono computati una volta sola**
- **Nei dischi moderni,** $t_{seek} = 3 \div 15$ **ms,** $t_{lat} = 2 \div 6$ **ms**
- ***Throughput* sino a circa 1 Gbit / sec**

Dischi ottici standard (CD)

- **Basati su tecnologia laser che consente di ottenere capacità e velocità di lettura/scrittura elevate**
- **CD-ROM (Compact Disk Read-Only Memory)**
  - **disco di plastica trasparente di 12 cm di diametro**
  - **al suo interno si trova un sottile disco di alluminio**
  - **le informazioni sono memorizzate come succes-sione di “buchi” nell'alluminio**
  - **ottenuti per stampa con macchinari industriali e poi letti da un laser**
  - **capacità sino a 700-800 Mbyte**

Dischi ottici scrivibili

- **CD-R (Compact Disk Registrabili)**
  - **si scrivono una volta sola – Write Once Read Many (WORM), tutto in una volta**
  - **si scrivono cambiando la riflettività di una strato di colorante posto sopra un riflettente metallico**
  - **si scrivono con lo stesso laser usato per leggerli, che altera il colorante dove vanno messi i "buchi"**
- **CD-RW (Compact Disk Read-Write)**
  - **scritti con la stessa tecnologia**
  - **la scrittura è reversibile (circa 1000 scritture)**
  - **anch'essi si scrivono tutti insieme, non a settori**

DVD e Blu-Ray

- I DVD (Digital Video Disc) sono dischi ottici di 2 generazione, con elevata densità:
  - 4,7 GB, ma anche 8.5 GB e 9.4 GB
  - DVD-Video e -Audio per film e musica
    - crittografati ma “cracked”
  - DVD-R, DVD-RW
  - HD DVD: da 15 GB a 45 GB
- Blu-Ray Disc (BD) proposto dalla Sony nel 2002 per film ad alta risoluzione
  - usa un laser blu, con lunghezza d'onda corta
  - 54 GB, ma già in uso 100 GB
  - crittografia AES per la protezione dei dati

“Dischi” a stato solido

- **Usano supporti elettronici per memorizzare in modo permanente le informazioni**
- **Usano la tecnologia delle memorie flash EPROM**
  - **“Memorie” o “Chiavi” USB: sino a 256 GB**
  - **Unità a stato solido: sino a 2 TB**
  - **Circa 10 volte più costose degli HD**
  - **Nessuna parte in movimento, minor consumo**
  - **10-50 volte più veloci degli HD**
  - **Minor numero di RW per bit (da 10000 a 1 M)**
- **Interfaccia uguale a quella dei dischi (tracce, settori...) per compatibilità**

I file e la loro gestione

I File

- **Le *informazioni permanenti* (che non si perdono allo spegnimento della macchina) nei calcolatori sono memorizzate in *file* (fascicoli)**
- **Un file system è l'astrazione informatica di un archivio di documenti, coi dati contenuti in fascicoli, e questi in cartelle (e poi in faldoni, scaffali, stanze...)**
- **Il concetto di file è rivolto all'utente, ed è indipendente dal mezzo sul quale viene memorizzato**
- **Un file è una sequenza di bit (o meglio di Byte):**
  - **le sue dimensioni si misurano in Byte (KB, MB, GB)**
  - **il Byte è l'unità minima di accesso al file**
- **E' l'unità base di archiviazione**

File

- **Un file può essere:**
  - **creato**
  - **modificato**
  - **cancellato**
  - **copiato**
  - **spostato**
  - **eseguito (se è un programma o un insieme di instruzioni)**
- **I file sono contenuti in *cartelle* (o *indirizzari*, o *directory*, o *folder*) organizzate in modo gerarchico:**
  - **una cartella può contenere altre cartelle e file**
  - **c'è una cartella primaria *root* (radice) che contiene tutte le altre, a vari livelli, e quindi anche tutti i file**

File e cartelle

- **Ogni file, oltre che dal proprio nome, è caratterizzato da degli *attributi*, ad es.:**
  - **tipo: (eseguibile, file di comandi, testo, immagine jpg,...) – spesso codificato nell'*estensione***
  - **dimensione: numero di byte contenuti nel file**
  - **data e ora (di creazione e/o di modifica)**
  - **utente proprietario**
  - **diritti di accesso al file per il proprietario e gli altri utenti**
- **Il nome di un file o di una cartella è una sequenza di caratteri (alfanumerici e caratteri di interpunzione ammessi), univoco entro la cartella di appartenenza**
- **La parte di nome dopo l'ultimo punto si chiama *estensione*: `conti.txt`, `win.exe`, `DataBase.odp`**

Il File System

- **E' il sistema che gestisce i file:**
  - **fornendo un'interfaccia verso l'utente**
  - **gestendone la memorizzazione fisica efficiente sulle unità di memoria di massa**
- **All'utente mostra, su comando o con una finestra, la struttura gerarchica delle cartelle, permette di "navigarla" e di effettuare operazioni sui file**
- **A livello di sistema**
  - **mappa i file sui settori dei dischi, sa ritrovarli e gestirli**
  - **Sa gestire le modifiche ai file (che possono cambiarne le dimensioni) in modo efficiente**

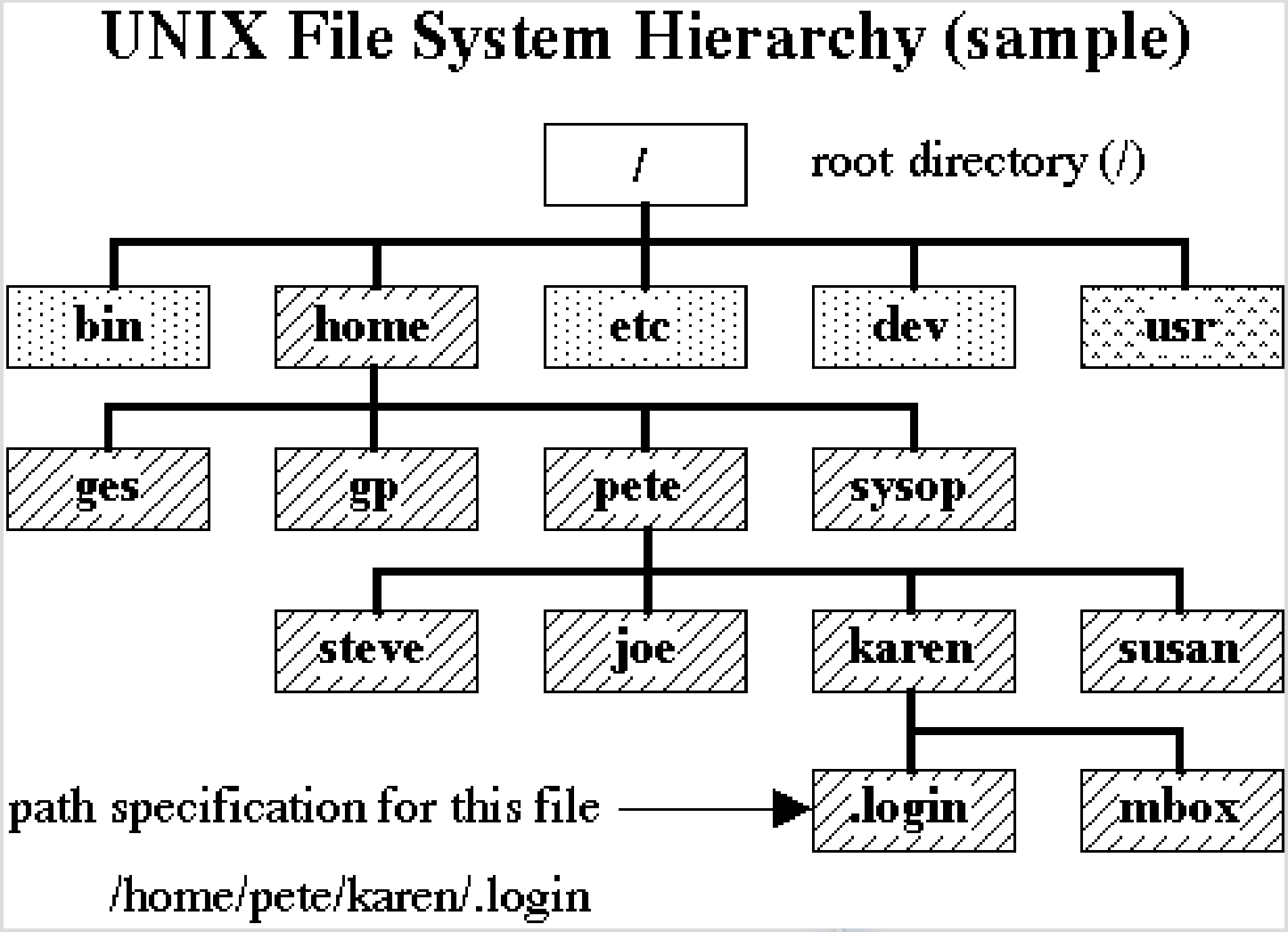
File System

- **Il Gestore del File System fa parte del sistema operativo ed è responsabile delle seguenti attività:**
  - **Creazione e cancellazione di file**
  - **Creazione e cancellazione di cartelle**
  - **Manipolazione di file e cartelle**
  - **Codifica del file system sulla memoria di massa**
- **Il gestore del file system deve garantire la correttezza e la coerenza delle informazioni**
- **Nei sistemi multi-utente, deve mettere a disposizione dei meccanismi di protezione che consentano agli utenti di proteggere i propri dati dall'accesso da parte di altri utenti non autorizzati**

Le cartelle e i file

- **La root è la cartella che contiene tutte le altre**
- **Ogni cartella può contenere altre cartelle**
- **Ogni computer ha un sistema di cartelle:**
  - **alcune facenti parte del sistema operativo**
  - **altre create e gestite dagli utenti**
- **Le cartelle formano quindi una *gerarchia***
- **Alla fine, le cartelle contengono i file, che sono le unità di informazione interessanti per gli utenti**
- **Ogni file è univocamente individuato dal proprio nome, e dalla lista dei nomi delle cartelle che lo contengono, sino alla root**

Grafo delle Cartelle (Linux)



Nomi dei file

- **Ogni file ha un nome unico entro la propria cartella**
- **Il nome contiene lettere, cifre e altri caratteri**
- **Minuscole e maiuscole sono considerate differenti (Unix, Linux) oppure no (DOS)**
- **La parte a destra dell'ultimo punto è l'*estensione***
- **Il *nome completo* è il nome di tutte le cartelle che contengono il file, a partire dalla root, sino al nome del file:**

  /home/michele/corsi/elemInfo/DataBase.odp

- **Qui il carattere separatore delle cartelle è '/' (Unix)**

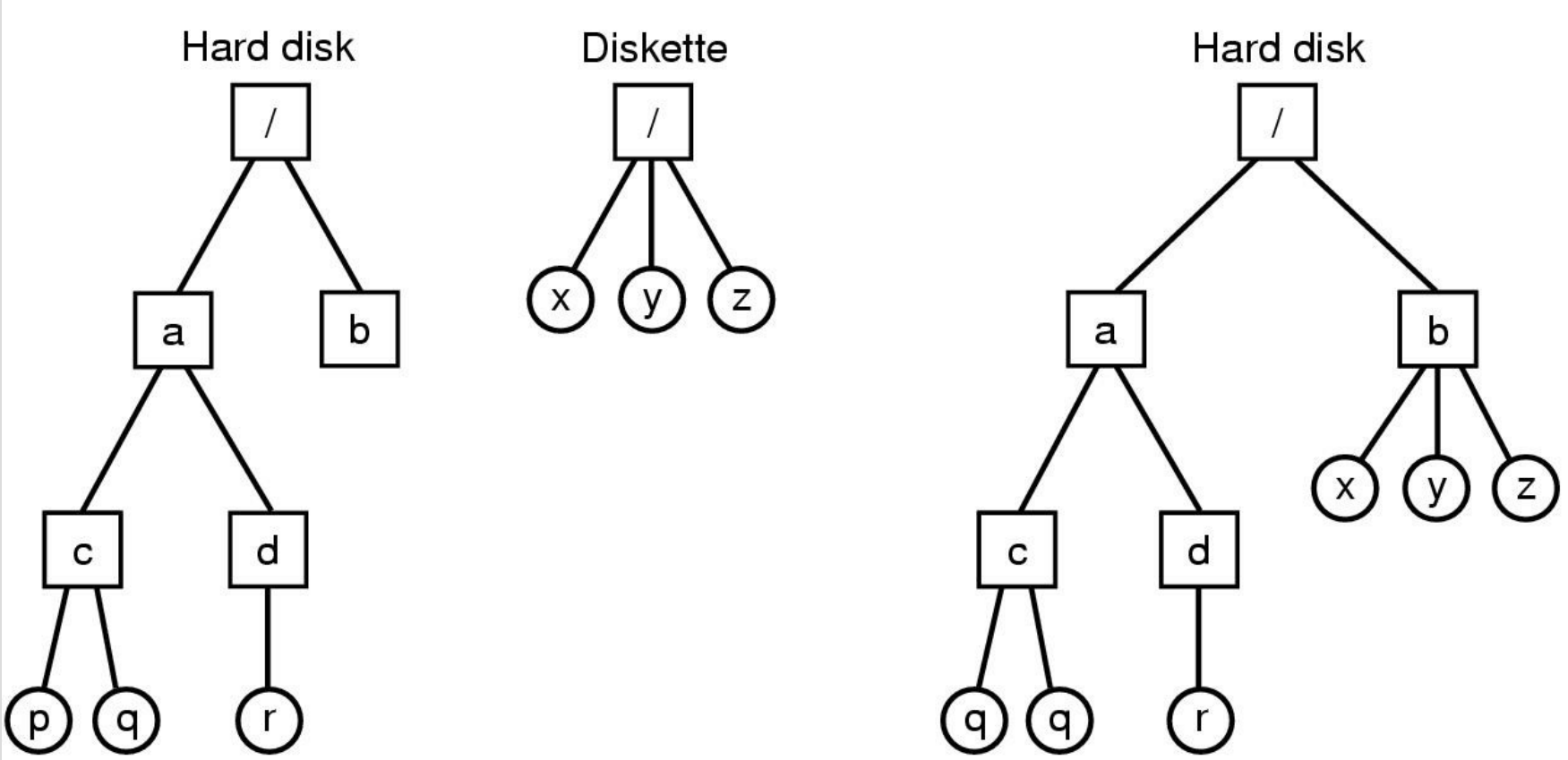
Ancora nomi

- **In Windows, il separatore è '\':**
  C:\Users\michele\corsi\elemInfo\DataBase.odp
- **Si noti che in Windows c'è una root per ogni *partizione* o *drive* in cui sono contenuti file (qui C:)**
- **Ad es., una penna USB corrisponde di solito al drive F: o G:**
- **In Unix, c'è una root unica del sistema; un nuovo drive è *montato* nella cartella: /media**
- **Ad es: /media/USB_Mio_Drive diventa la "root" dei file entro la penna USB**

Partizioni

- Un disco fisso può essere diviso in partizioni
- Ogni partizione contiene un file system
- Per creare una partizione, occorrono programmi appositi
- I dati dei settori della nuova partizione sono cancellati
- Vi sono vari tipi di file system: FAT, FAT32, NTFS, ext2, ext3, ext4, HFS Plus
- Differenti partizioni possono contenere FS di tipo diverso, eventualmente con diversi SO

Montaggio di un file system



prima del mount

dopo il mount

Cartella corrente

- **Esiste una *cartella corrente* (c.c.) del sistema**
- **In Linux è la cartella: /home/nome_user**
- **Se un file non ha nome completo, si assume che sia entro la cartella corrente**
- **Così: pippo.txt è in realtà (in Linux):**

  /home/nome_user/pippo.txt
- **Un nome sintetico della c.c è il punto**
- **Il doppio punto ".." è la cartella che contiene la c.c.**

  ./pippo.txt ≡ /home/nome_user/pippo.txt

  .. ≡ /home → ../altro_user ≡ /home/altro_user

Contenuto di un file

- **Per il S.O., un file contiene una sequenza di byte**
- **Applicazioni specifiche sanno interpretare e usare il contenuto (es. file .odp in LibreOffice/Open Office)**
- **I file sono letti e scritti in *blocchi*, tipicamente di 1,2 KB**
- **Fisicamente, il contenuto di un file (i blocchi) può essere contiguo sul disco, o "spezzato" – per l'utente ciò è trasparente**
- **Il S.O. mette a disposizione funzioni per leggere, scrivere e posizionarsi entro un file (posizione corrente)**
- **L'unità minima di accesso è il byte**

Tipiche operazioni su un file

- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get Position
- Set Position
- Get Attributes
- Set Attributes
- Rename
- Copy

Le tipiche cartelle di Unix/Linux

- / root: la cartella che contiene tutte le altre
- /bin contiene gli eseguibili base del sistema
- /etc contiene i file di configurazione
- /dev contiene file di interscambio con le periferiche
- /home contiene le “home” degli utenti
- /lib contiene le librerie di sistema
- /media contiene i FS delle memorie rimovibili
- /opt contiene programmi aggiuntivi
- /tmp contiene file temporanei, cancellati ogni tanto
- /usr contiene librerie e programmi non critici o installati dagli utenti
- /var contiene file che possono cambiare spesso

Comandi di accesso ai file (Unix)

>pwd	Mostra il nome della cartella corrente
>ls	Mostra il contenuto della cartella corrente
>ls -l	Come ls, ma mostra informazioni sui file
>cd ..	Si sposta sulla cartella che contiene quella corrente (che diviene la corrente)
>cd *cart*	Si sposta su ***cart*** (che diviene la cart. corrente)
>rm *file*	Cancella il file di nome dato
>mkdir *cart*	Crea la cartella di nome dato
>rmdir *cart*	Cancella la cartella di nome dato
>**cp *f1 f2***	Copia il file di nome ***f1*** in quello di nome ***f2***

Semplici organizzazioni interne dei file

Contenuto di un file

- **Vi sono vari modi per organizzare un file:**
  - **file sequenziale: si legge e si scrive tutto in una volta**
    - **ad es: un file di testo, un file .odp**
  - **file con "tag": file sequenziale in cui le informazioni sono separate da "tag" (etichette):**
    - **il lettore interpreta i tag e gestisce in modo opportuno le informazioni tra i tag**
    - **ad es: file HTML, file XML**
  - **file con record: il file contiene sequenze di byte di lunghezza fissa (*record*)**
    - **si possono leggere/scrivere record singoli**
    - **l'accesso si chiama *random* (casuale)**

Esempio: file HTML

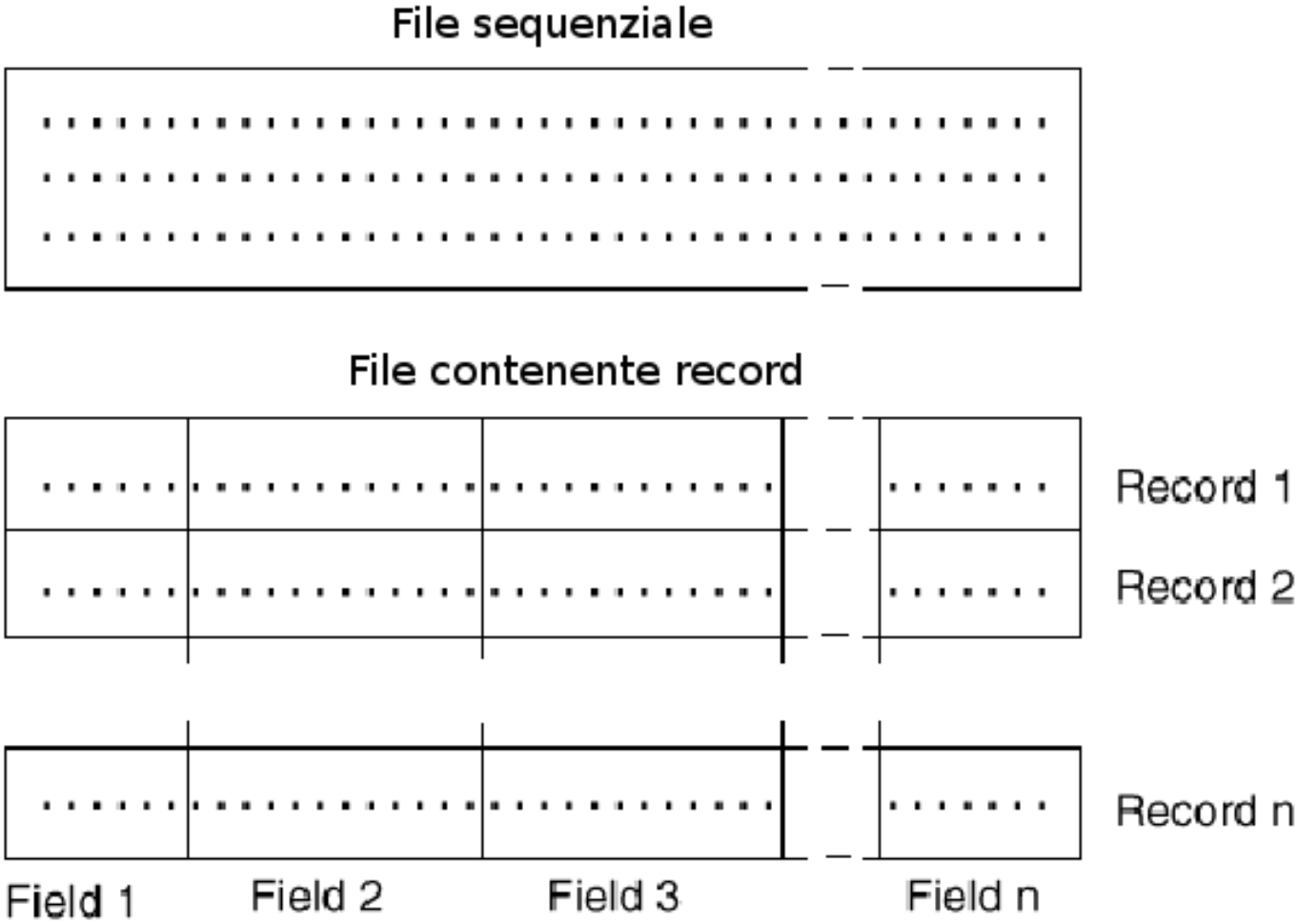
```
<html>
<head>
<title>unica.it - Università degli studi di Cagliari.
</title>
<meta name="keywords" content="unica.it - Università degli
studi di Cagliari.">
 <link rel="stylesheet" href="/UserFiles/File/css2/css.css"
type="text/css" media="all" />
<!--[if IE]><link rel="stylesheet"
href="/UserFiles/File/css2/ie7.css" type="text/css"
media="all" charset="utf-8" /><![endif]-->
</head>

<body class="front">
<div class="wrapper">
....
```

File con record

- **Un file con record può essere semplicemente un "blocco" di *N* record uguali**
  - **se la lunghezza del record è *L*, il file contiene *NL byte***
  - **se i record vanno da 0 a *N* – 1, il byte di inizio del record *k*-esimo è *kL***
  - **conoscendo la posizione del record, la sua lettura e scrittura sono immediate**
- **Il problema di un file con semplici record è che occorre conoscere l'associazione tra record e posizione *k***
- **Altrimenti, almeno all'inizio occorre leggere tutti i record in modo sequenziale**

File con record



File con record

- **Ogni record include dell'informazione di controllo (ad es. un bit che denota se il record è vuoto o no)**
- **All'inizio ad es., tutti i record potrebbero essere vuoti, e poi scritti poco a poco**
- **Oltre all'info di controllo, i record contengono una sequenza di *campi di lunghezza fissa*, che a loro volta contengono le vere informazioni**
  - **ad es: Nome, Cognome, anno nascita, ecc.**
- **Occorre sovradimensionare i campi, per non avere problemi (ad es., nomi cognomi troppo lunghi)**
- **Dello spazio è necessariamente sprecato (campi non pieni, campi non valorizzati, record vuoti)**

Accesso a file con record

- **Di solito, si accede a un record conoscendone già parte dell'informazione (es. cognome e nome, CF)**
- **Tale informazione è detta anche *chiave* del record**
- **In tal caso, si può accedere al record in modo veloce:**
  - **stabilendo i campi della chiave**
  - **ordinando i record in ordine alfabetico sulla chiave**
  - **in tal modo, la lettura di un record data la chiave può avvenire con una ricerca binaria (molto efficiente)**
  - **in caso di inserzione di nuovi record, occorre inserirli nella giusta posizione, spostando tutti i record già presenti (molto inefficiente)**
  - **in pratica, si usano strutture dati complesse (alberi)**

File con record ordinati su un campo

Campo di ordinamento

|
|

Name	EID	Address	Birthdate	Salary
Adams, John				
Alan, Delon				
Bill, Clinton				
. . . .				
Jay, Shana				
Mary Ann, Smith				
Melinda, Perkin				
. . . .				
Nancy, Davies				
Son, Nguyen				

File hash

- **Sono un modo molto efficiente per accedere ai record data la chiave**
- **Non hanno però ordinamento alfabetico**
- **Occorre stabilire a priori una stima del numero di record N', e dimensionare il file ad es. a $N = 1{,}4\ N'$ record (40% di spazio in più)**
- **Si usa una funzione hash, H che associa alla chiave un intero tra 0 e N-1: $H(key) : S_{key} \rightarrow [0,N\text{-}1]$**
- **Ad es., se la chiave è una stringa di 16 caratteri (CF) allora $S_{key} = [0, 2^{128}\text{-}1]$ (16 car. sono 128 bit)**
- **Gli intervalli $[0,M] \subset \mathbb{N}$**

File hash

- **Data una chiave *x*, *H*(*x*) è la posizione del record corrispondente nel file, $H(x) \in [0, N-1]$**
  - **se il record in posizione *H*(*x*) è vuoto → un record di chiave data non è presente nel file**
  - **se il record in posizione *H*(*x*) è pieno → lo leggo con un solo accesso**
  - **se voglio scrivere il record, ne ho la posizione *H*(*x*)**
  - **se alla posizione *H*(*x*) c'è un record vuoto, scrivo quello nuovo e marco che è pieno nell'informazione di controllo**
  - **se alla posizione *H*(*x*) c'è un record pieno:**
    - **se questo ha la stessa chiave *x*, lo sovrascrivo: non posso avere due record con la stessa chiave!**

File hash: inserimento con collisione

- **Data una chiave *x*, *H*(*x*) è la posizione del record corrispondente nel file**
  - **se alla posizione *H*(*x*) c'è un record pieno con una chiave diversa da *x*, ho una *collisione* :**
    - **scrivo il nuovo record nel primo spazio libero in una zona (o in un file) di *overflow*, al di fuori dello spazio con gli *N* record**
    - **"collego" il nuovo record a quello presente in posizione *H*(*x*):**
      - **imposto l'informazione di controllo del record in posizione *H*(*x*) a "collisione"**
      - **inserisco nella stessa informazione di controllo l'indirizzo del record di overflow**

File hash: inserimento con doppia collisione

- se alla posizione $H(x)$ c'è un record pieno con una chiave diversa da x, e se tale record già marca una collisione:
  - accedo al record collegato nella zona di overflow
  - se tale record ha chiave uguale a x, lo sovrascrivo
  - altrimenti, scrivo il nuovo record nella zona di *overflow*,
  - "collego" il nuovo record a quello già presente nella zona di overflow
  - ho così una *catena* di un record nella zona principale, e due record nella zona di overflow
- il procedimento può essere iterato...

Esempio: dati fiscali

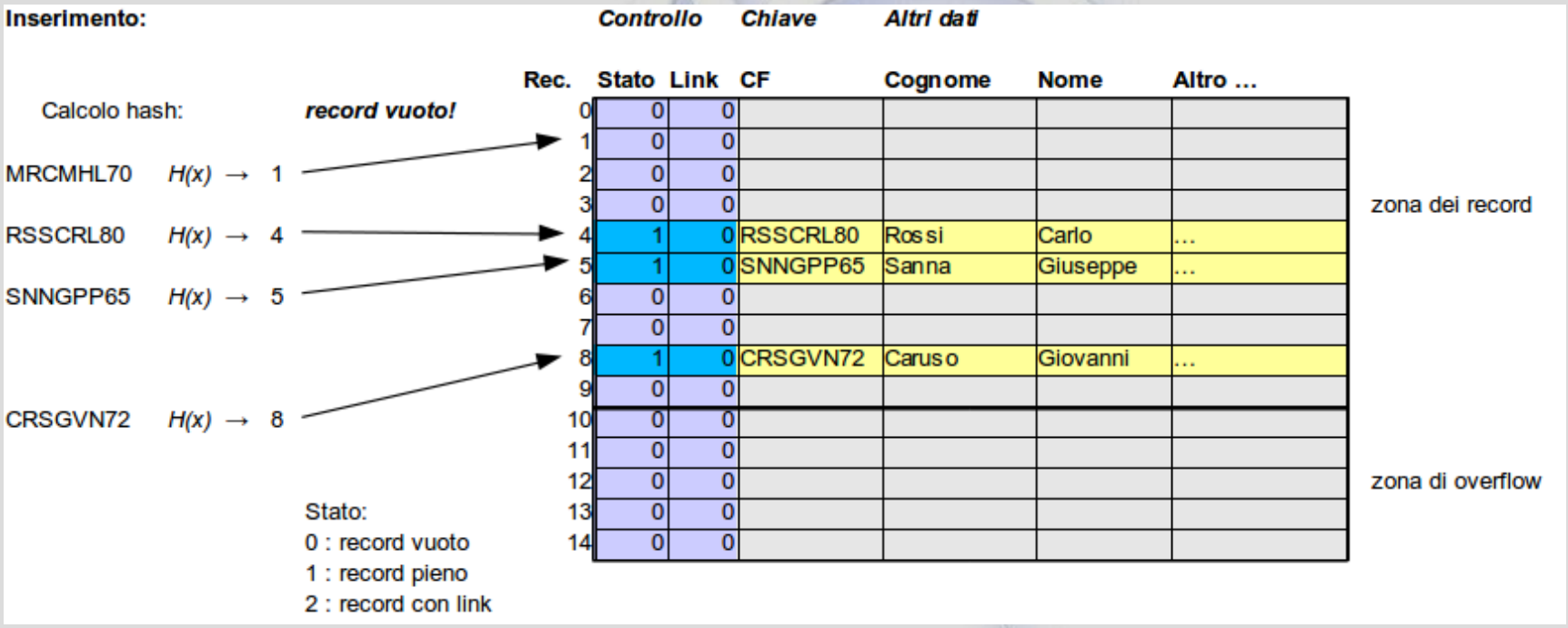
- **Voglio accedere in modo efficiente ai record dei contribuenti italiani, dato il codice fiscale (CF)**
- **Record: dati anagrafici e fiscali del contribuente**
- **Controllo: 6 byte per contenere indirizzo di record con chiavi dello stesso hash nello spazio di overflow**
- **Dati anagrafici (in chiaro):**
  - **CF, nome, cognome, sesso, data, luogo nascita**
  - **indirizzo, recapiti**
- **Dati fiscali (codificati in byte):**
  - **anno (1B) , reddito (4B), 6 voci reddito (24B), pagamenti (4B) = 33B x 10 anni (330 B)**
- **Ipotizziamo 6 B + 176 B + 330 B = 512 B a testa**

Esempio: dati fiscali

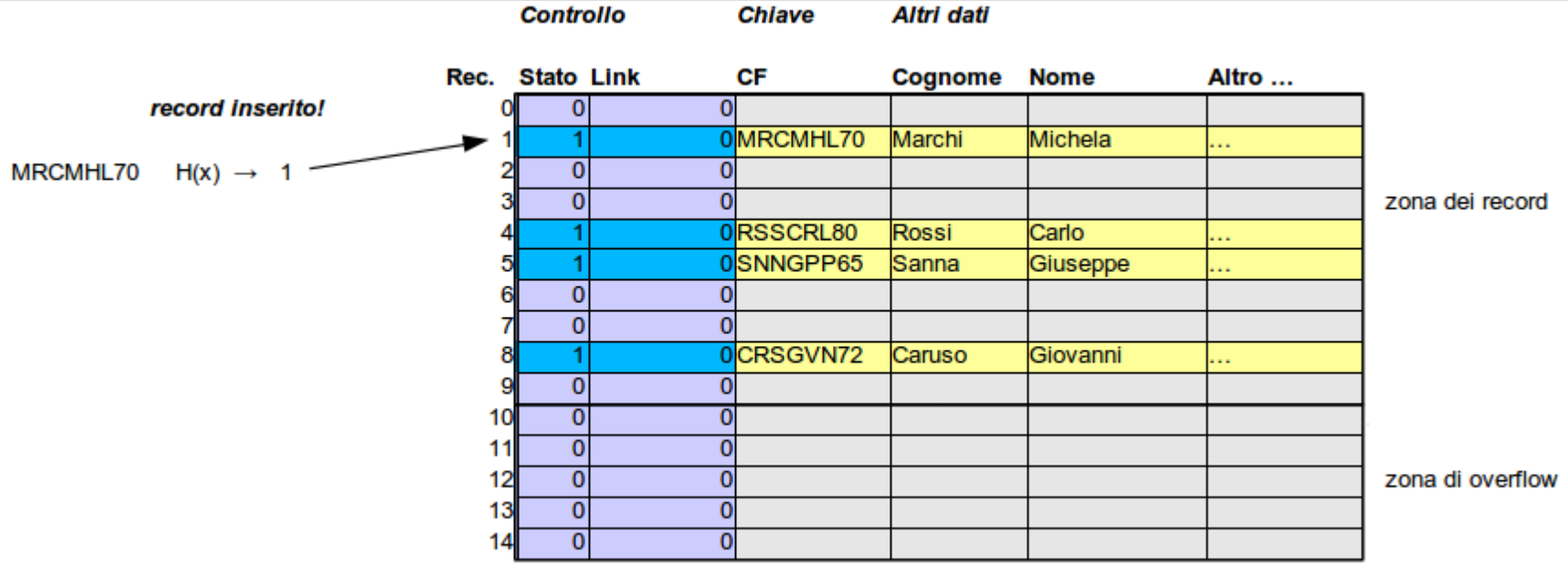
- Ipotizziamo $N' = 20$ milioni di contribuenti fisici
  - $1{,}4\ N' = N = 28.000.000$ record → file di ≃ 14 GB
  - aggiungiamo lo spazio di overflow → 30M record
  - so che $log_2\ (28.000.000) \simeq 25$
  - spazio della chiave (CF): 16 x 8 = 128 bit
  - Segue che: $H(x) : [0, 2^{128}\text{-}1] \rightarrow [0, 2^{25}]$
- Nel lucido seguente è riportato un esempio semplificato, con:
  - chiave x di 8 car.
  - spazio di indirizzamento di 10 record: [0, 9]
    → $H(x) : S_{key} \rightarrow [0, 9]$
  - spazio di overflow di 5 record

Esempio: dati fiscali (semplificato)

- **Nel file mostrato, voglio inserire un nuovo record, di chiave: MRCMHL70**
- **Il record corrispondente all'hash della chiave è vuoto:**

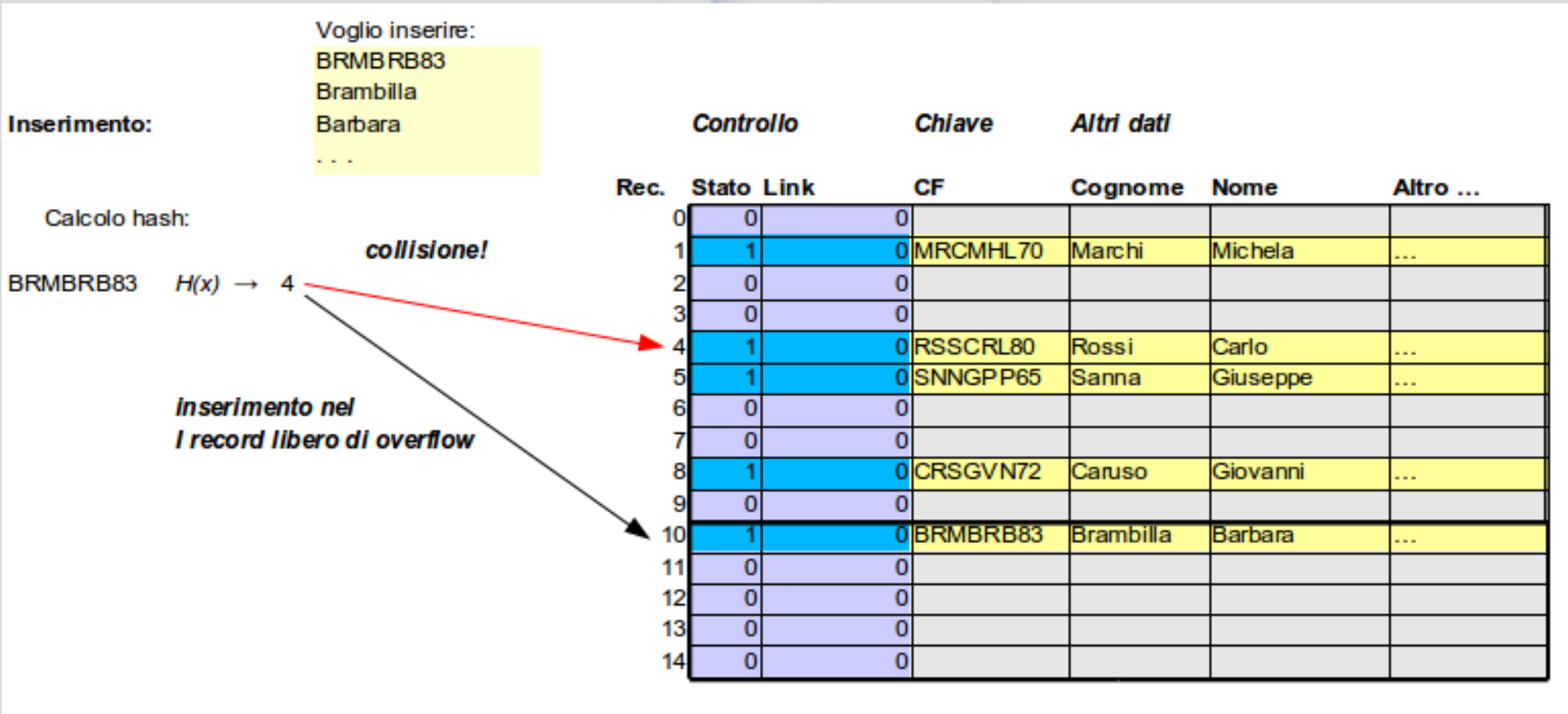


Inserimento in record vuoto



Inserimento con collisione

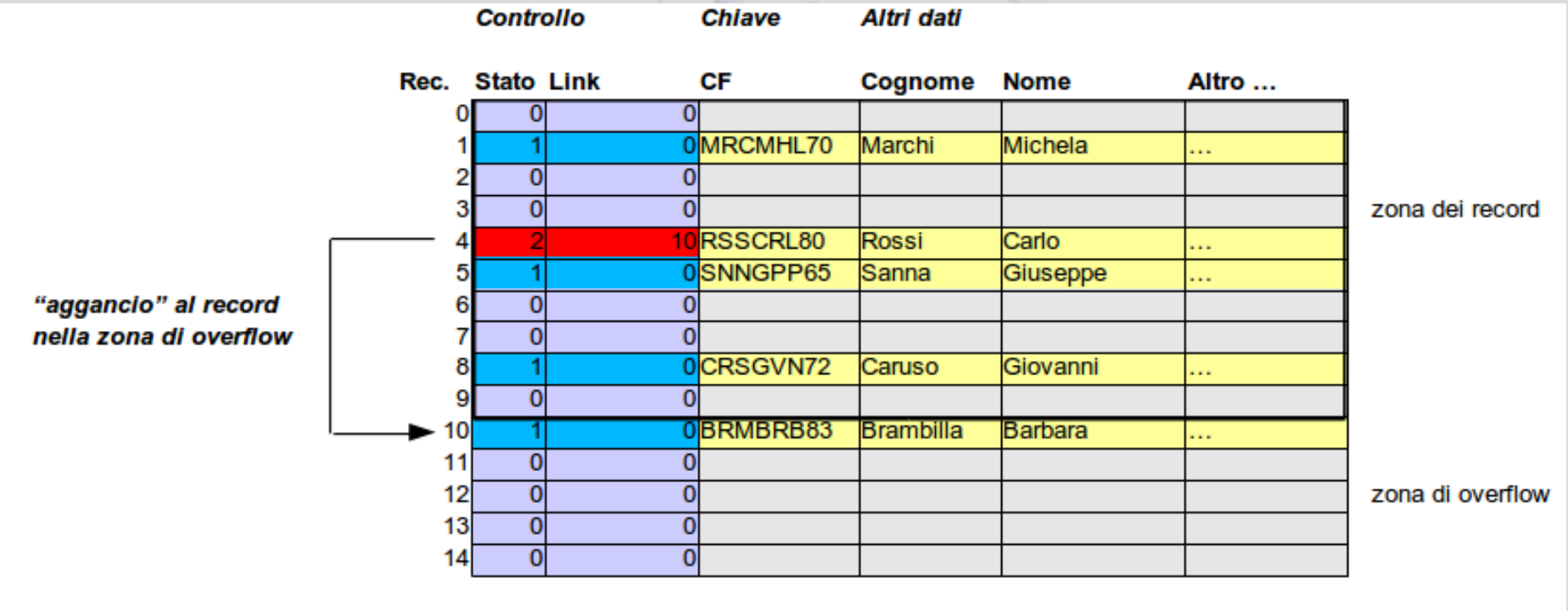
- Voglio ora inserire un nuovo record, di chiave: BRMBRB83
- Il record corrispondente all'hash della chiave è occupato dal record di chiave: RSSCRL80
- Inserisco il nuovo record nell'area di overflow (record 10):



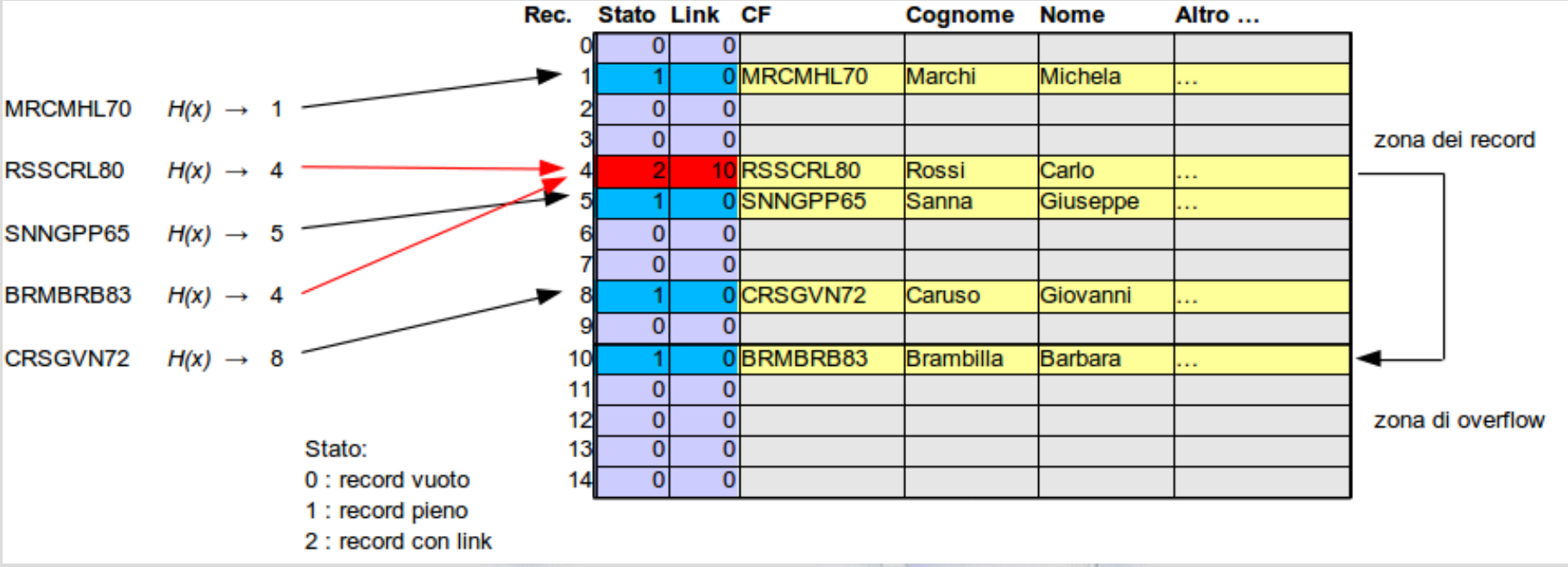
	Controllo		Chiave	Altri dati		
Rec.	Stato	Link	CF	Cognome	Nome	Altro ...
0	0	0				
1	1	0	MRCMHL70	Marchi	Michela	...
2	0	0				
3	0	0				
4	1	0	RSSCRL80	Rossi	Carlo	...
5	1	0	SNNGPP65	Sanna	Giuseppe	...
6	0	0				
7	0	0				
8	1	0	CRSGVN72	Caruso	Giovanni	...
9	0	0				
10	1	0	BRMBRB83	Brambilla	Barbara	...
11	0	0				
12	0	0				
13	0	0				
14	0	0				

Inserimento con collisione (2)

- Completo l'operazione “agganciando” il nuovo record a quello nella zona dei record:
  - pongo il suo stato a 2
  - pongo l'indirizzo del nuovo record (10) come “Link”



Lo stato del file dopo le inserzioni



Rec.	Stato	Link	CF	Cognome	Nome	Altro ...
0	0	0				
1	1	0	MRCMHL70	Marchi	Michela	...
2	0	0				
3	0	0				
4	2	10	RSSCRL80	Rossi	Carlo	...
5	1	0	SNNGPP65	Sanna	Giuseppe	...
6	0	0				
7	0	0				
8	1	0	CRSGVN72	Caruso	Giovanni	...
9	0	0				
10	1	0	BRMBRB83	Brambilla	Barbara	...
11	0	0				
12	0	0				
13	0	0				
14	0	0				

Dopo un'altra inserzione con doppia collisione

- Ipotizziamo di aver inserito un altro record con H(x) = 4
  - c'è collisione sia col record 4 che col record 10
  - il nuovo record è inserito in posizione 11 e collegato al 10: