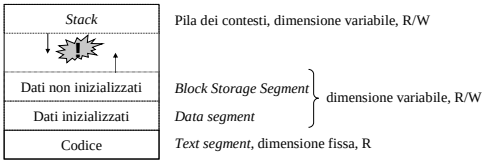
Gestione della memoria – 1

- Massima semplicità per massima portabilità su diverse architetture fisiche
- Ogni processo possiede un proprio spazio di indirizzamento virtuale privato
  - Suddiviso in 4 sezioni



Gestione della memoria – 2

- Il segmento dati varia in dimensione a seconda delle richieste del programma (cf. `malloc()` di C)
  - POSIX non definisce queste chiamate di sistema
  - Una parte del segmento dati può ospitare *file* mappati in memoria
- Il segmento *stack* contiene l'ambiente d'esecuzione corrente (record di attivazione) e cresce in direzione opposta al segmento dati
- Il segmento codice può essere condiviso tra più processi
  - Gli altri segmenti no, tranne che per processi clonati da una `fork()`

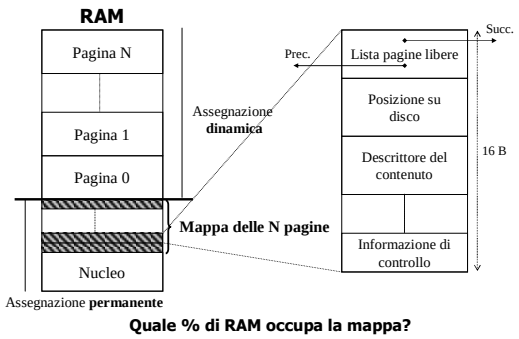
Gestione della memoria (UNIX) – 3

- In origine l'allocazione di memoria principale avveniva mediante ***swap*** di processi
  - Rimpiazzo di interi processi (del loro spazio di indirizzamento) quando una particolare esecuzione rilevava mancanza di memoria
    - A seguito di `fork()`, di allocazione esplicita da programma, o di allocazione implicita da chiamata di procedura
  - Il gestore (*swapper*) creava lo spazio necessario salvando su disco i processi sospesi con più tempo d'esecuzione recente e minor priorità
    - Bastava utilizzare una lista dei blocchi liberi su disco

Gestione della memoria (UNIX) – 4

- In seguito, fu introdotta paginazione con modalità **a richiesta** (***paging on demand***)
  - Un processo è eseguibile se il suo descrittore e la sua tabella delle pagine si trovano in RAM
    - Il suo spazio di indirizzamento è caricato da disco, per pagine per ogni riferimento che richiede dati non ancora in RAM
    - Nessun caricamento anticipato di pagine (no *working set*)
  - Il processo 2 (*page daemon*) gestisce lo stato delle pagine in RAM
    - Nucleo di S/O e mappa delle pagine (*core map*) sempre in RAM
    - Il resto è paginato e ciascuna pagina indica il proprio uso
      - Codice, dati, *stack*, tabella delle pagine (altrimenti in lista pagine libere)

Mappa delle pagine (*core map*)

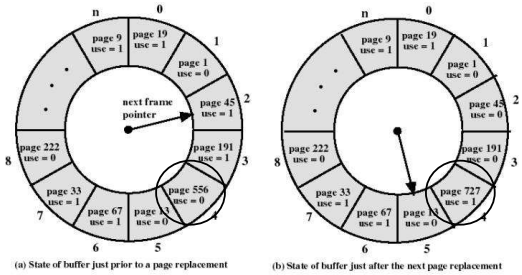


Quale % di RAM occupa la mappa?

Gestione della memoria (UNIX) – 5

- *Page daemon* verifica periodicamente (@ 1/4 s) che vi siano ≥ `lotsfree` pagine libere
  - Se ne mancano ne libera quante ne servono salvandone il contenuto corrente su un'area di disco
  - La selezione delle pagine in uscita usa un algoritmo "a doppia passata" (***two-handed clock algorithm***)
    - Lista circolare delle pagine
    - La 1ª passata pone a 0 il *bit* di riferimento
    - La 2ª passata, a distanza programmabile, rimuove le pagine nel frattempo non riferite (*bit* vale 1 altrimenti)

Clock algorithm (a una passata)



Intervallo troppo lungo tra 2 passate successive!

Gestione della memoria (UNIX) – 6

- *Page daemon* limita anche la frequenza di paginazione spostando alcuni processi su disco
  - Quelli che non abbiano eseguito negli ultimi 20 s
  - Tra i 4 più grandi, quello da più tempo in memoria
- Se vi è spazio libero *page daemon* riporta in RAM processi pronti selezionati con una euristica di "valore"
  - Caricando solo il descrittore di processo e la sua tabella delle pagine
    - Il resto viene caricato su base di *paging on demand*

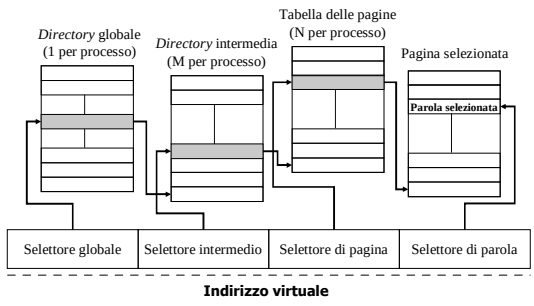
Gestione della memoria (GNU/Linux) – 6

- Lo spazio di indirizzamento virtuale di processo è ampio 4 GB (per architetture a 32 *bit*)
  - 1 GB riservato (e non visibile in *modo normale*) per la sua tabella delle pagine e altri dati di controllo ad uso del nucleo
- Spazio suddiviso in **regioni** = sequenze contigue di pagine
  - Le regioni possono essere **non consecutive** tra loro
- Ogni regione ha un descrittore noto al nucleo
  - Lo spazio di indirizzamento virtuale di un processo in effetti è visto come una lista di descrittori

Gestione della memoria (GNU/Linux) – 7

- La `fork()` di GNU/Linux replica per il figlio l'intera lista di descrittori del padre
- Le pagine del figlio sono fisicamente duplicate solo in caso di modifica (***copy on write***)
  - La regione è marcata **R/W**
  - Le sue pagine (dati) sono inizialmente marcate **R**
  - Ogni richiesta di scrittura causa eccezione così il nucleo duplica la pagina richiesta e marca la copia come **R/W**

Gestione della memoria (GNU/Linux) – 8



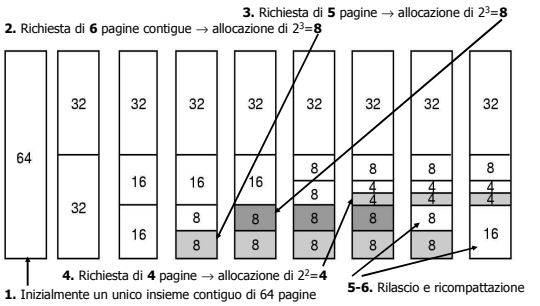
Gestione della memoria (GNU/Linux) – 9

- Il nucleo rimane sempre in RAM
  - Di dimensione **variabile** a causa del caricamento dinamico di moduli di gestione dispositivi
- La RAM rimanente viene usata per
  - **[P1]** Le pagine attive dei processi utente
  - **[P2]** Una *cache* di blocchi di *file* usata dal *file system*
    - Dimensione **variabile** organizzata per pagine
  - **[P3]** Un insieme di pagine utente inattive ma presenti
- La RAM viene assegnata in porzioni di dimensione variabile e arbitraria

Gestione della memoria (GNU/Linux) – 10

- **Algoritmo di allocazione primario** (*buddy*)
  - Ogni richiesta di ampiezza N è arrotondata a $2^n \geq N$
  - La memoria disponibile viene frazionata in metà successive fino a frazioni di ampiezza 2^n
    - 1 frazione viene assegnata al richiedente
    - Una struttura ausiliaria contiene la testa di liste predefinite di frazioni di ampiezza 2^i (i=0, …, n) per velocizzare la ricerca
  - La memoria disponibile usata per l'allocazione è sempre la frazione libera di minore dimensione
  - Al rilascio ogni frazione tornata libera si unisce con la frazione vicina se libera (il suo *buddy*)
  - Due algoritmi sussidiari cercano di ridurre la frammentazione causata dall'algoritmo primario

Gestione della memoria (GNU/Linux)
Funzionamento del buddy algorithm



Gestione della memoria (GNU/Linux) – 11

- I segmenti codice e i *file* mappati in memoria hanno un corrispondente *file* su disco
- Al resto (aree di lavoro dei processi) si assegna una partizione paginata (***paging partition***) vista come *byte stream* oppure un *file* di pagine
  - La partizione paginata è d'uso più semplice e veloce
    - Nessun limite fissato di scrittura, scrittura contigua
- Le pagine libere sulla partizione e/o sul *file* sono individuate mediante *bitmap*
  - Lo spazio disponibile viene assegnato alle pagine di lavoro rimosse temporaneamente dalla RAM

Gestione della memoria (GNU/Linux) – 12

- **kswapd** è il *page daemon*, con periodo 1 s
  - Per ogni attivazione, esegue fino a 6 passate di un ciclo di lavoro cercando pagine da spostare su disco
    - Tra quelle **[P2]** e **[P3]**, con una variante del *clock algorithm*
    - Tra quelle condivise da più processi ma poco usate
    - Tra quelle **[P1]**
      - Cominciando dal processo con più pagine, scandendo l'intera lista dei suoi descrittori di regione (per indirizzo virtuale) con *clock algorithm* ma solo sulle pagine attive
  - Ogni pagina selezionata modificata
    - Posta in attesa di riscrittura se ha corrispondente su disco
    - Altrimenti salvata su *paging partition* o *paging file*
- Il *daemon* **bdflush**, gestisce la riscrittura

Gestione dell'I/O – 1

- UNIX tratta i dispositivi di I/O come *file* di tipo speciale, ciascun con posizione specifica nel FS (**/dev/…**)
  - *File* orientati a carattere (p.es.: tastiera, rete, …)
  - *File* orientati a blocco (p.es.: disco)
- Un gestore (*device driver*) è associato in modo esclusivo a ciascun dispositivo (o famiglia di dispositivi dello stesso tipo)
  - Una coppia di indici <maggiore, minore> identifica precisamente ciascun dispositivo di I/O

Gestione dell'I/O – 2

- GNU/Linux consente caricamento dinamico dei moduli gestore di (nuovi) dispositivi
  - Soluzione preferibile alla configurazione statica che richiede ricompilazione dell'intero nucleo
    - Particolarmente a fronte di grande varietà di *hardware*
- Il caricamento dinamico richiede varie azioni di configurazione
  - Rilocazione dello spazio di indirizzamento del modulo
  - Allocazione delle risorse necessarie (p.es.: interruzione assegnata al dispositivo)
  - Configurazione del vettore delle interruzioni
  - Attivazione ed inizializzazione del gestore

Gestione dell'I/O – 3

- Uno speciale *file* (detto *socket*) per la connessione di rete ed il suo protocollo
  - Può essere creato e distrutto dinamicamente
  - Un *socket* è associato a uno specifico indirizzo di rete
- 3 tipi di connessione con scelta alla creazione
  - Connessione affidabile a flusso di caratteri (~ **TCP**)
    - Il gestore garantisce la correttezza della trasmissione
    - Invio e ricezione per blocchi di dimensione variabile
  - Connessione affidabile a flusso di pacchetti (**TCP**)
    - Come sopra, ma con invio e ricezione solo per pacchetti
  - Trasmissione inaffidabile di pacchetti (**UDP**)
    - L'utente deve occuparsi di trattare gli eventuali errori

Gestione dell'I/O – 4

- Una zona di RAM è usata come *cache* dedicata per velocizzare R/W di dati su dispositivi a blocchi
  - Ogni blocco richiesto in lettura viene prima cercato in *cache*
  - Ogni blocco scritto viene trattenuto in *cache* il più a lungo possibile
    - Fin quando la *cache* è piena e serve spazio
    - Fino all'attivazione del *daemon* di scrittura su disco