Introduzione a Linux

Lezione 4
Gestione dei processi

Ruggero Donida Labati

Laboratorio di Sistemi Operativi
Università degli Studi di Milano
Dipartimento di Informatica
A.A. 2024/2025

Panoramica della lezione

- Verrà introdotta brevemente la gestione della memoria in Linux
- Verranno illustrate le caratteristiche dei processi
- Verranno introdotti i metodi per la gestione dei processi

Sommario

1. La memoria in Linux
2. I processi in Linux
   - Tipi di processi
   - Struttura dei processi
   - Gestione dei processi
3. Redirezione dell'Input/Output
4. Esercizi

1. La memoria in Linux

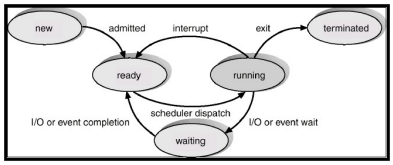
1. Introduzione alla memoria in Linux
2. Evoluzione e integrazione dei processi
3. Memoria allocabile
4. Configurazione «Swappiness»
5. Approfondimenti

Introduzione

- Linux è un sistema operativo multi-user e multi-tasking
  - Numerosi processi in esecuzione contemporaneamente
- Implementa meccanismi per la gestione di numerosi processi in memoria
  - Allocazione spazio di memoria per ogni processo
  - Protezione degli indirizzi
  - Paginazione
  - Memoria virtuale

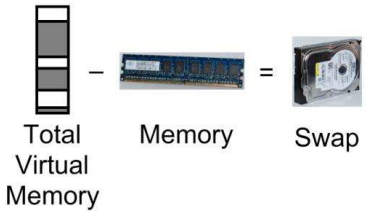
Evoluzione e integrazione tra i processi

- Sono presenti meccanismi per controllare l'evoluzione e l'integrazione tra i processi
  - Avvio
  - Sospensione
  - Duplicazione
  - Terminazione
  - Ecc.



Memoria allocabile

- La memoria allocabile è limitata dallo spazio di indirizzamento della CPU
  - 4 GB su macchine a 32 bit
- Quando la memoria allocata supera la memoria fisica, viene utilizzata la partizione di swap

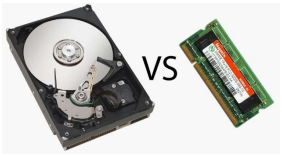


Configurazione «Swappiness» (1/2)

- Lo spazio di swap viene usato per estendere la memoria utilizzabile oltre la capacità fisica
  - Spazio di memoria dato dalla somma di memoria fisica + swap
  - I processi utilizzati meno frequentemente vengono spostati nell'area di swap
    - HD di molti ordini di grandezza più lenti della RAM
- Il parametro «swappiness» definisce quanto spesso Linux sposterà contenuti dalla RAM allo swap
  - Default: 60
  - Swap inizia quando la memoria è usata al ~40%

Configurazione «Swappiness» (2/2)

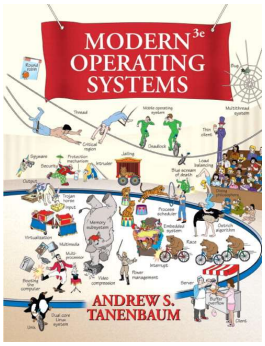
- È possibile configurare il parametro
  - Es. diminuire per sistemi con molta RAM (no macchina virtuale….)
  - Valore attuale:
    - cat /proc/sys/vm/swappiness
  - Cambiare il valore
    - Es. 10
    - sudo bash -c "echo 'vm.swappiness = 10' >> /etc/sysctl.conf"
    - sudo sysctl -p



Approfondimenti

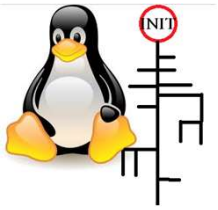
- La gestione della memoria nei sistemi operativi è un argomento complesso

- Approfondimenti
  - Corso di sistemi operativi
  - *Andrew S. Tanenbaum – Modern Operating Systems, 3° edition*



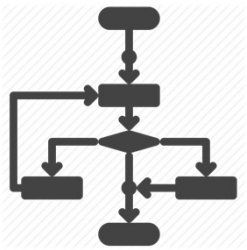
2. I processi in Linux

1. Tipi di processi
2. Struttura dei processi
3. Gestione dei processi



Tipi di processi

1. Processi interattivi
2. Processi automatici
3. Daemon (o servizi)



Processi interattivi (1/2)

- Sono avviati manualmente dall'utente tramite un terminale
- Possono essere in foreground o in background
  - In foreground bloccano il terminale finchè l'esecuzione non è completata
  - In background permettono l'uso del terminale per altri processi

Processi interattivi (2/2)

- È possibile avviare un processo interattivo in background aggiungendo *&*
  - Es. Apertura di un nuovo terminale da shell
  - *xterm &*
  - Se si avviasse *xterm* in foreground la prima shell rimarrebbe inutilizzabile
- Comando per visualizzare i processi in background
  - *jobs*

Processi automatici

- I processi automatici (o batch) non sono associati ad un terminale
- Sono accodati dal sistema operativo ed eseguiti con base FIFO
- Sono eseguiti in due situazioni
  - Ad un'ora predefinita
    - Comando *at*
  - Quando il carico del sistema è < 80 %
    - Comando *batch*

Daemon

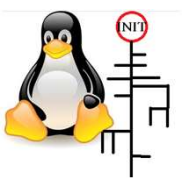
- Processi che sono sempre in esecuzione
- Inizializzati in fase di avvio della macchina
- In genere rimangono in background finchè non sono richiesti
  - *Networking*
  - *HTTP server*
  - *Ecc.*
- È possibile fermarli o riavviarli (es. per applicare modifiche alla loro configurazione)

Struttura dei processi

1. Creazione
2. Terminazione
3. Sospensione e ripristino
4. Attributi

Creazione di un processo

- Un processo viene creato da un altro processo utilizzando due chiamate di sistema
  - Duplicazione (*fork*)
  - Sovrascrittura del codice con il codice del nuovo processo (*exec*)
- Il processo che origina tutti gli altri è il processo *init*
- Il processo *init* rimane il processo padre dei daemon



Terminazione di un processo

- Quando un processo termina restituisce un codice di uscita (valore di ritorno)
  - Derivato dal linguaggio C
- I codici di uscita cambiano per ogni programma
  - Documentati nel manuale associato al programma
  - *man <nome_programma>*
- Il comando *kill* permette di terminare un processo
  - *kill <ID_processo>*

Sospensione e ripristino di un processo

- Per sospendere un processo avviato in foreground bisogna inviare il segnale di SIGTSP
  - *Ctrl + Z*
- Portare in foreground un processo che era in background
  - *fg <nome_processo>*
- riprendere in background l'esecuzione di un job sospeso
  - *bg <nome_processo>*

Attributi di un processo

- ID del processo (*PID*)
- ID del processo padre (*PPID*)
- Terminale a cui è associato (*TTY*)
- Utente reale (*RUID*) e utente effettivo (*EUID*)
  - Utente reale è chi avvia il processo
  - Utente effettivo controlla l'accesso alle risorse
  - Sono in genere uguali
- Gruppo reale (*RGID*) e gruppo effettivo (*EGID*)
- Numero *nice*
  - Controlla la priorità del processo

Gestione dei processi

1. Elenco dei processi
2. Tempo di esecuzione di un processo
3. Priorità dei processi
4. Schedulazione dei processi

Elenco dei processi (1/2)

- Lista dei processi che attualmente si trovano in esecuzione
  - *ps*
- In realtà ogni utente riceve il riepilogo dei processi che stanno girando con i suoi permessi
- Lista di tutti i processi presenti sulla macchina
  - *ps -aux*

Elenco dei processi (2/2)

- Lista dei processi e relazioni tra di loro
  - *pstree*
- Lista dei processi aggiornata in modo dinamico
  - *top*

Tempo di esecuzione di un processo

- Tempo di esecuzione di un processo
  - *time <nome_processo>*
- Output
  - Tempo effettivo trascorso (*real*)
  - Tempo di CPU utilizzato dal processo (*user*)
  - Tempo di CPU utilizzato dal sistema per il processo (*sys*)

Priorità dei processi (1/2)

- La priorità di un processo è definita dal suo numero *nice*
  - Un alto valore di *nice* corrisponde ad una bassa priorità
  - Un basso valore di *nice* significa che il processo è importante
- È possibile diminuire il valore di nice dei processi in base alle necessità
  - Utile per processi che usano molto la CPU
  - Poco utile per processi legati all'I/O
    - Automaticamente impostati con un basso valore di *nice*

Priorità dei processi (2/2)

- Definire la priorità di un processo tramite il suo valore nice
  - Comando *nice*
  - *nice <nome_processo>*

Schedulazione dei processi

1. Tipi di schedulazione
2. Schedulazione tramite *sleep*
3. Schedulazione tramite *cron* e *anacron*
4. Schedulazione tramite *at*
5. Schedulazione tramite *batch*

Tipi di schedulazione (1/2)

- La schedulazione dei processi permette di impostarne l'esecuzione in date e orari stabiliti
- Tre tipi di schedulazione
  - Posticipare l'esecuzione di una certa quantità di tempo
  - Schedulare l'avvio di un processo in modo regolare
  - Impostare l'avvio di un processo in un certo momento

Tipi di schedulazione (2/2)

- Cinque strumenti possibili
  - *sleep*
  - *cron*
  - *anacron*
  - *at*
  - *batch*

Schedulazione tramite *sleep*

- Il comando *sleep* permette di ritardare l'esecuzione di un processo
  - Es. carico di sistema troppo elevato
    - *(sleep 10000; <processo_pesante>) &*
  - Promemoria
    - *(sleep 1800; echo "Lunch time..") &*

Schedulazione tramite *cron* (1/4)

- Il comando *cron* permette di configurazione il servizio *crond*
  - Esegue i comandi in momenti determinati
  - Configurazione contenuta nel file *crontab*
- La configurazione è testuale
  - Modificando il file *crontab*
    - *crontab -e*

Schedulazione tramite *cron* (2/4)

- Struttura dei file *crontab*
  - le righe vuote e quelle che cominciano con il simbolo # vengono ignorate
  - le righe possono contenere un assegnamento ad una variabile d'ambiente o un comando *cron*

- Formato generico dei comandi *cron*
  - *data-orario comando*
  - *data-orario* si scompone in altri cinque campi:
    - *minuti ore giorni-del-mese mesi giorni-della-settimana*
  - I campi possono contenere un asterisco (*)
    - Ogni valore possibile

Schedulazione tramite *cron* (3/4)

- I campi possono essere impostati in diversi modi
  - Valori singoli
    - Ogni volta che l'orologio raggiunge quel valore
  - Intervalli
    - Ogni volta che l'orologio raggiunge un valore compreso nell'intervallo
  - Elenchi
    - Ogni volta che l'orologio raggiunge uno dei valori possibili
  - Passi
    - Ogni volta che è trascorsa l'unità di tempo specificata nel passo

Schedulazione tramite *cron* (4/4)

- Esempio di comando *cron*
  - Ogni giovedì alle 16:38 invia un'email a pippo con oggetto «calcetto»
    - *38 16 * * 3 mail –s "calcetto" pippo*

Schedulazione tramite *anacron* (1/2)

- Strumento simile a *cron*, utilizzato nel caso di macchine non accesse sempre
  - Configurato tramite il file */etc/anacrontab*
- Due tipi di istruzioni
  - Assegnamento a variabili d'ambiente
  - comandi
- I comandi sono nella forma
  - *n-giorni n-secondi-ritardo nome-attribuito-al-job comando*

Schedulazione tramite *anacron* (2/2)

- Opzioni
  - *n-giorni*
    - Indica la cadenza con cui deve essere eseguito il comando
  - *n-secondi-ritardo*
    - Indica il numero di secondi che deve essere atteso prima di cominciare
  - *nome-attribuito-al-job*
    - Attribuisce un nome al job
  - *comando*
    - Comando da eseguire.

Schedulazione tramite *at* (1/2)

- Processo da eseguire una sola volta

- Comandi a terminale
  - *at <tempo_di_esecuzione>*
    at> <comando_1>
    at> <comando_2>
    Ecc.
    Ctrl + D

Schedulazione tramite *at* (2/2)

- È possibile usare un file con la lista dei comandi da eseguire
  - *at -f <file> <ora>*
  - Es. *at -f routine 13:30 + 3 days*
  - Esegue i comandi contenuti nel file "routine" fra tre giorni alle 13:30.

Schedulazione tramite *batch*

- Simile a *at*
- I processi sono eseguiti evitando di eccedere il carico di sistema impostato
  - Non è obbligatorio specificare un orario
- Comando a terminale
  - *batch*
  - *at> <comando_1>*
  - *at> <comando_2>*
  - Ecc.
  - Ctrl + D

3. Redirezione dell'Input / Output

1. Canali di comunicazione del sistema
2. Pipe
3. Redirezione su file

Canali di comunicazione del sistema

- I sistemi operativi in generale hanno tre canali di comunicazione
  - Standard input
    - Il canale tramite il quale vengono ricevuti i dati da processare
    - Normalmente è usata la tastiera
  - Standard output
    - Il canale tramite il quale vengono emessi i dati processati
    - Normalmente è usato il terminale
  - Standard error
    - È il canale tramite il quale vengono riportati gli errori
    - Normalmente è usato il terminale

Pipe

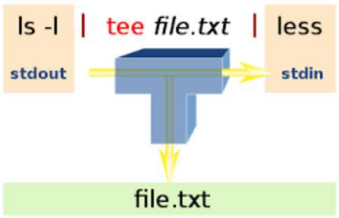
- Il comando pipe | permette di usare lo standard output di un processo come standard input di un altro processo
  - *<programma_1> | <programma_2>*
- Es. utile per ordinare in modo alfabetico l'output di un comando
  - Ordinare in modo alfabetico l'elenco delle directory
    - *ls | sort*

Redirezione su file (1/2)

- Può essere utile salvare l'output di un programma su file (redirezione dello standard output)
  - Comando >
  - *«programma» > «file_output.txt»*
  - *Il comando >> aggiunge l'output al termine del file*
- Oppure usare un file come input di un programma (redirezione dello standard input)
  - Comando <
  - *«programma» < «file_input.txt»*

Redirezione su file (2/2)

- Redirezione dello standard error
  - Comando *2>*
  - *«programma» 2> «file_output.txt»*



In sintesi

1. La memoria in Linux
2. I processi in Linux
3. Redirezione dell'Input/Output

4. Esercizi (1/5)

- Cambiate il valore di «swappiness» da 60 a 10
- Provate i seguenti comandi e interpretate il risultato:
  - ps
  - top
  - sleep 5
  - jobs

4. Esercizi (2/5)

- Eseguite «sleep 15» in foreground, sospendetelo e quindi mettetelo in background. Alla fine riportatelo in foreground
  - Provate anche con il comando «xterm»
- Eseguite «sleep 15» in background, quindi terminate il processo
- Eseguite «sleep 15» in background, sospendetelo e quindi riprendete in foreground l'esecuzione del processo

4. Esercizi (3/5)

- Eseguite «sleep 300», effettuare il logoff e il login
  - Il processo è ancora in esecuzione?
- Usate cron per mandare un messaggio a voi stessi ogni minuto
  - Disabilitatelo poi (fastidioso)

4. Esercizi (4/5)

- Elencate in senso decrescente i file all'interno della cartella personale
- Salvate l'output di un processo su file
- Usate il comando «pipe» | in combinazione con «grep» per estrarre il testo dallo standard output

4. Esercizi (5/5)

- Usate il comando «time» per scoprire i tempi di esecuzione dei comandi
- Fermate il daemon che gestisce la rete, quindi avviatelo nuovamente