

Lezione 2 – Creazione e terminazione dei processi

Sistemi Operativi I

Modulo 3 - Gestione del processore

Unità didattica 1 - Processi

Vincenzo Piuri

Università degli Studi di Milano - SSRI - CDL ONLINE

Sommario

- Modelli computazionali
- Modalità e funzioni per
 - creazione e attivazione dei processi
 - terminazione dei processi

Processi come flussi di operazioni

- Processo = flusso di esecuzione di computazione
- Flussi separati = processi separati
 - Come si generano?
 - Come un programma si trasforma in un insieme di processi?
 - Come interagiscono i processi?
- Flussi sincronizzati
 - processi evolvono sincronizzandosi
- Flussi indipendenti
 - processi evolvono autonomamente

Modellazione della computazione a processi

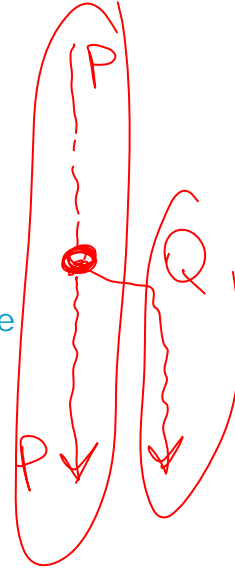
- Modelli di computazione:
 - Processo monolitico
 - Processi cooperanti
- Modelli di realizzazione del codice eseguibile:
 - Programma monolitico
 - Programmi separati
- Realizzazione dei modelli di computazione:
 - Programma monolitico è eseguito come processo monolitico
 - Programma monolitico genera processi cooperanti
 - Programmi separati sono eseguiti come processi cooperanti (ed eventualmente generano ulteriori processi cooperanti)

Generazione di un processo

- Il processo in esecuzione chiama una funzione di sistema operativo che crea e attiva un nuovo processo

fork

- Processo generante → processo padre
- Processo generato → processo figlio

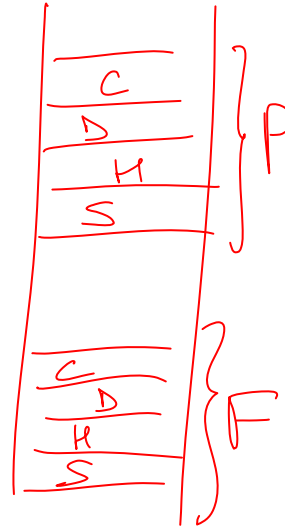


Risorse dei processi

- Condivise col padre
- Parzialmente condivise col padre
- Indipendenti dal padre (ottenute dal sistema)
- Passaggio dei dati di inizializzazione

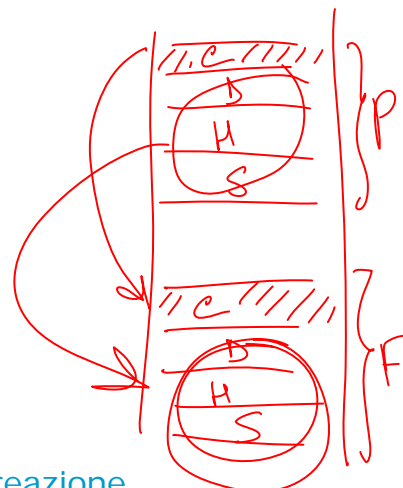
Spazio di indirizzamento (1)

- Lo spazio di indirizzamento del processo figlio è sempre **distinto** da quello del processo padre



Spazio di indirizzamento (2)

- Spazio del processo figlio è il duplicato dello spazio del processo padre



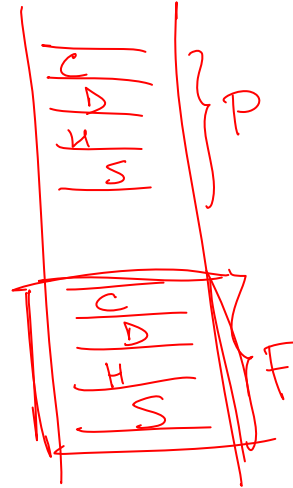
- stesso programma
- stessi dati all'atto della creazione

Spazio di indirizzamento (3)

- Il processo figlio ha un nuovo contenuto del suo spazio di indirizzamento

exec

- nuovo programma



Esempio

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int pid;

    /* genera un altro processo */
    pid = fork();

    if (pid < 0) { /* si è verificato un errore */
        fprintf(stderr, "Fork fallita");
        exit(-1);
    }
    else if (pid == 0) { /* processo figlio */
        execlp("/bin/ls", "ls", NULL);
    }
}
```

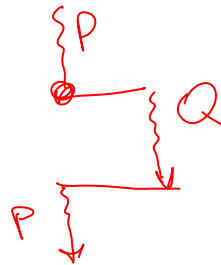
Esecuzione dei processi

- Il padre continua l'esecuzione in modo concorrente ai figli

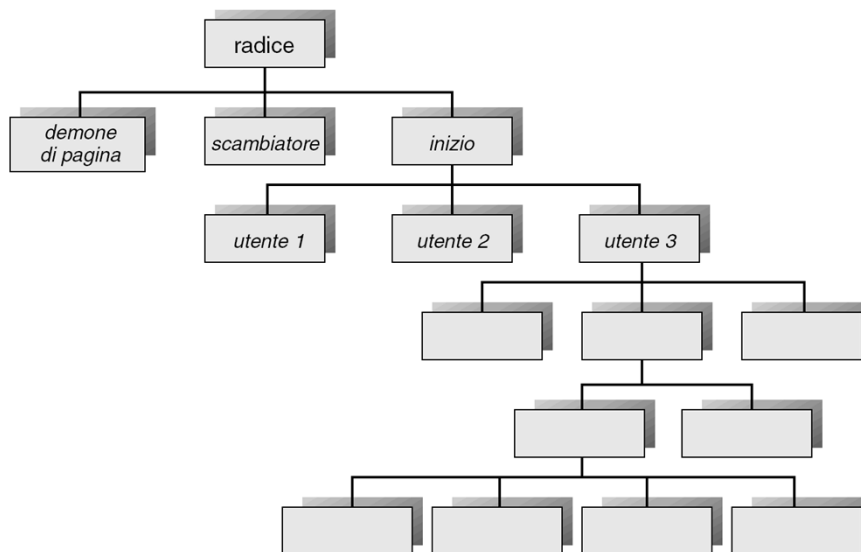


- Il padre attende finché alcuni o tutti i suoi figli sono terminati

Wait



Albero dei processi



Terminazione di un processo (1)

Terminazione normale dopo l'ultima istruzione

exit

- restituzione un valore di stato
- deallocazione delle risorse

```
}  
else if (pid == 0) { /* processo figlio */  
    execlp("/bin/ls", "ls", NULL);  
}  
else { /* processo padre */  
    /* il processo padre attenderà il completamento del figlio */  
    wait(NULL);  
    printf("Figlio terminato");  
    exit(0);  
}  
}
```

Terminazione di un processo (2)

Terminazione in caso di anomalia

abort

Cause possibili

- Eccessivo uso di una risorsa
- Compito non più necessario
- Terminazione a cascata

In sintesi

- Caratteristiche e funzioni dell'attivazione e della terminazione dei processi

