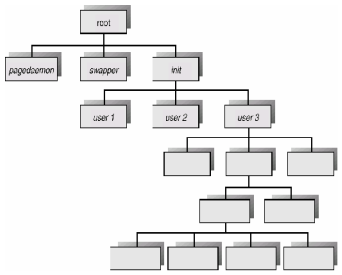
Creazione di un processo…

Un processo (***padre***) crea altri processi (***figli***) che, a loro volta, possono creare altri processi, il tutto a formare un ***albero di processi***



Esempio: UNIX

… e scenari possibili

- ***Spazio degli indirizzi***
  - Il figlio ***duplica*** quello del padre e non lo modifica
  - Il figlio esegue un ***nuovo codice***
- ***Condivisione delle risorse***
  - Il padre e il figlio condividono ***tutte*** le risorse
  - Il figlio condivide ***una parte*** delle risorse del padre
  - Il padre e il figlio ***non*** condividono ***alcuna*** risorsa
- ***Sincronizzazione***
  - Vanno in esecuzione ***in concorrenza***
  - Il ***padre attende*** la terminazione del ***figlio***

La system call "*fork*" di UNIX …

fork crea un nuovo processo che ha una copia dello spazio di indirizzi del padre:

- ripartono dalla stessa istruzione dopo la fork, cioe' si origina ***un nuovo program counter*** all'interno dello stesso codice
- per poterli distinguere il ***codice di ritorno*** della chiamata alla fork e' ***differente*** tra il padre e il figlio

… e un esempio d'uso

```
while (TRUE) {                                  /* repeat forever*/
   type_prompt( );                              /* display prompt*/
   read_command (command, parameters)           /* input from
    terminal*/

if (fork() != 0) {                              /* fork off child
   process*/
   /* Parent code*/
   waitpid( -1, &status, 0);                    /* wait for child to exit
   */
} else {
   /* Child code*/
   execve (command, parameters, 0);             /* execute command*/
}
```

exec usata dopo una **fork** rimpiazza lo spazio di memoria con un nuovo programma

wait permette al padre di attendere la terminazione del figlio e quindi, eventualmente, di ricevere risultati dal figlio

Terminazione "normale" di un processo

Il processo esegue l'ultima istruzione e poi chiede a OS di terminare (mediante una istruzione di ***exit***)

Tutte le risorse richieste o detenute dal processo vengono deallocate da OS

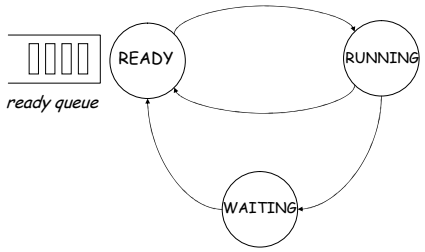
Terminazione "anormale" di un processo

Il padre puo' terminare l'esecuzione di uno dei figli (mediante una istruzione di ***abort***)

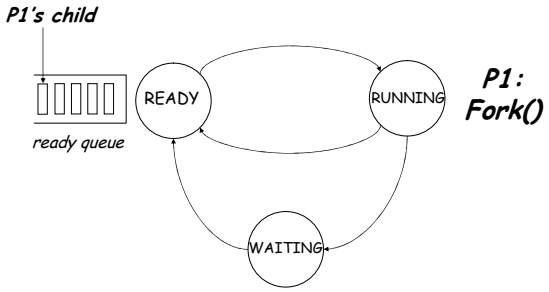
- perche' -

- Il figlio ha ecceduto nell'uso delle risorse (in tal caso il padre deve poter controllare lo stato dei figli)
- Il compito assegnato al figlio non e' piu' richiesto
- Il padre sta terminando (terminazione automatica dei figli in alcuni OS)

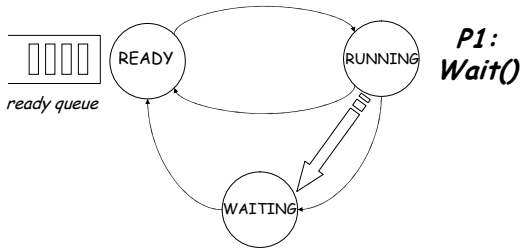
Riassumendo sull'esecuzione di processi…



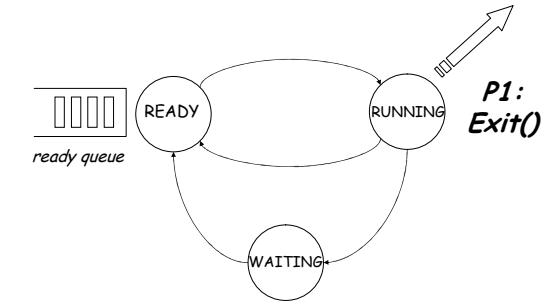
Riassumendo sull'esecuzione di processi…



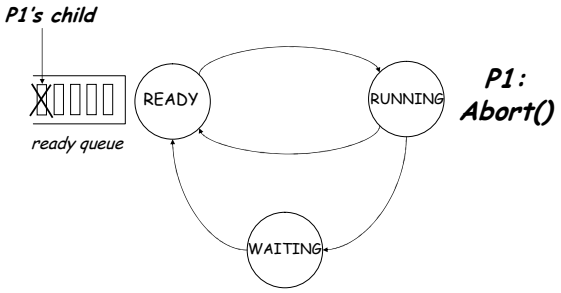
Riassumendo sull'esecuzione di processi…



Riassumendo sull'esecuzione di processi…



Riassumendo sull'esecuzione di processi…



Cooperazione tra processi

I processi possono ***influenzare*** l'esecuzione l'uno dell'altro

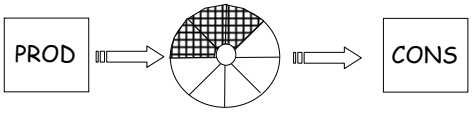
Vantaggi della cooperazione

- Condivisione di informazioni
- Velocizzazione del calcolo
- Modularita' di compiti
- Convenienza dell'utente

Per regolare questa "influenza" OS ha bisogno di meccanismi di ***comunicazione*** e di ***sincronizzazione***

Una semplificazione del problema della cooperazione: il problema *Produttore-Consumatore*

Il processo ***produttore*** produce informazioni che sono consumate da un processo ***consumatore***



Con un ***buffer di taglia finita*** bisogna regolamentare le operazioni di produzione e di consumo in maniera "opportuna"

Una soluzione : memoria condivisa

Dichiarazione strutture dati

var *n* ;
type *item* = … ;
var *buffer* : **array** [0… *n* –1]
of *item* ;
in, out: 0… *n* –1;

Processo ***produttore***

repeat
…
produce an item in *nextp*
…
while (*in* +1) **mod** *n* = *out*
do *no-op;*

buffer [*in*] := *nextp* ;
in := (*in* + 1) **mod** *n* ;

until *false;*

Processo ***consumatore***

repeat

while *in* = *out* **do** *no-op;*

nextc := *buffer* [*out*];
out := (*out* +1) **mod** *n* ;

…
consume the item in *nextc*
…
until *false;*

Questione aperta

Con questa soluzione si riescono a utilizzare al piu' ***n-1*** locazioni del buffer condiviso

⇩

Trovare una soluzione che permetta di utilizzare tutte le ***n*** locazioni del buffer condiviso (?!)

Ruolo di OS nella cooperazione

Nella soluzione del Produttore-Consumatore i due processi potevano condividere un buffer e il loro codice era stato ***scritto appositamente*** per quel tipo di struttura

E' compito di OS mettere in grado, in maniera ***trasparente***, i processi di ***scambiarsi informazioni tra loro***

Interprocess Communication (IPC)

IPC (parte di OS destinata a gestire la comunicazione) mette a disposizione dei processi utente essenzialmente due system calls :

- ***send***
- ***receive***