Sistemi Operativi - Tutoraggi

Laurea in Ingegneria Informatica
Università Tor Vergata
Tutor: Romolo Marotta
Docente del corso: Francesco Quaglia

Processi

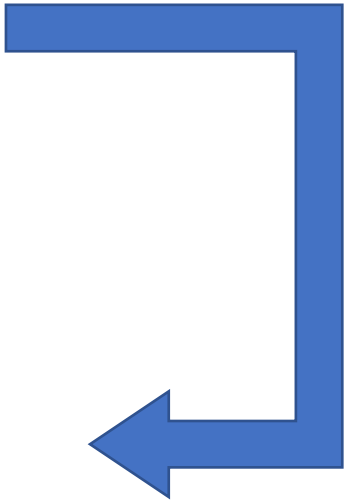
1. fork, wait, exit, exec
2. CreateProcess, WaitForSingleObject, ExitProcess, GetExitCodeProcess

(Alcuni) Servizi di sistema per gestire i processi

Creare un processo
Permettere ad un processo di terminare la propria esecuzione
Attendere la terminazione di processo figlio

(Alcuni) Servizi di sistema per gestire i processi

DESCRIZIONE	UNIX/LINUX
Creare un processo	fork
Permettere ad un processo di terminare la propria esecuzione	exit
Attendere la terminazione di processo figlio	wait



Standard POSIX:
https://pubs.opengroup.org/onlinepubs/9699919799/functions/fork.html

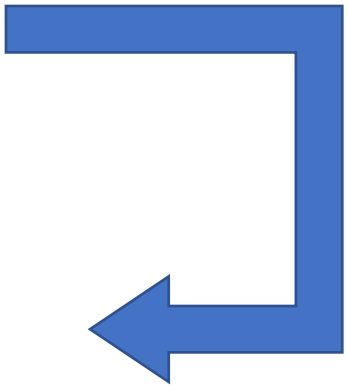
NAME
fork - create a new process

SYNOPSIS
```
#include <unistd.h>
pid_t fork(void);
```

DESCRIPTION
The fork() function shall create a new process.
*The new process (child process) shall be an **exact copy** of the calling process (parent process) except as detailed below:*
- *The child process shall have a unique process ID.*
- *…*

(Alcuni) Servizi di sistema per gestire i processi

DESCRIZIONE	UNIX/LINUX
Creare un processo	fork
Permettere ad un processo di terminare la propria esecuzione	exit
Attendere la terminazione di processo figlio	wait



Standard POSIX:
https://pubs.opengroup.org/onlinepubs/9699919799/functions/exit.html

NAME
fork - create a new process

SYNOPSIS
```
#include <stdlib.h>
void exit(int status);
```

DESCRIPTION
The value of status may be 0, EXIT_SUCCESS, EXIT_FAILURE,or any other value, though only the least significant 8 bits (that is, status & 0377) shall be available from wait().

……..
Finally, the process shall be terminated ….

(Alcuni) Servizi di sistema per gestire i processi

DESCRIZIONE	UNIX/LINUX
Creare un processo	fork
Permettere ad un processo di terminare la propria esecuzione	exit
Attendere la terminazione di processo figlio	wait

Standard POSIX:
https://pubs.opengroup.org/onlinepubs/9699919799/functions/wait.html

NAME
fork - create a new process

SYNOPSIS
```
#include <sys/wait.h>
pid_t
wait(int *status_location);
```

DESCRIPTION
The wait() … functions shall obtain status information … pertaining to one of the caller's child processes. The wait() function obtains status information for process termination from any child process.

……

The wait() function shall cause the calling thread to become blocked until status information generated by child process termination is made available….

Esempio 1

Esercizio 1

- Scrivere un programma in C che prende inizialmente una stringa da input (può contenere anche spazi bianchi) e la salva in un buffer
- fork-are un processo figlio che manda in stampa la stessa stringa acquisita dal processo padre.
- Il processo padre termina solo dopo che il processo figlio ha terminato (verificare che tale ordine è rispettato stampando i PID dei processi).

Esercizio 2

- Scrivere un programma in C che prende inizialmente una stringa da input (può contenere anche spazi bianchi) e la salva in un buffer
- fork-are 2 processi figli che contribuiscono a stampare la stringa inversa della stringa acquisita dal processo padre.
- Il processo padre termina solo dopo che i processi figli hanno terminato.

Sostituzione di programma

- Meccanismo per sostituire il programma associato al corrente processo di esecuzione
- Famiglia di funzioni **exec** permettono di definire:
  - il programma che sostituirà il codice del processo corrente
  - dove cercare il programma corrente (p)
  - i parametri da passare al programma come parametri multipli (l) o come array (v)
  - l'ambiente del nuovo processo (e)

```
SYNOPSIS
#include <unistd.h>
int execl(const char *pathname, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(const char *pathname, const char *arg, ..., char *const envp[] */);
int execv(const char *pathname, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execvpe(const char *file, char *const argv[], char *const envp[]);
```

Esercizio 3

- Scrivere un programma in C che prende inizialmente N (a piacere) stringhe rappresentanti N directory corrette
- fork-a quindi N processi che andranno ad eseguire il comando *ls* su una directory differente.
- Il processo padre termina dopo i processi figli

Creazione di Processi in Win32

- CreateProcess ≈ fork+exec
- ExitProcess ≈ exit
- WaitForSingleObject ≠ wait/waitpid
  - Change notification
  - Console input
  - Event
  - Memory resource notification
  - Mutex
  - Process
  - Semaphore
  - Thread
  - Waitable timer

Creazione di Processi in Win32

- CreateProcess ≈ fork+exec
- ExitProcess ≈ exit
- WaitForSingleObject ≠ wait/waitpid
- GetExitCodeProcess
  - IS_ERROR
  - HRESULT_CODE

Esercizi

- Svolgere gli esercizi 1,2 e 3 in ambiente Windows