

Esercitazione

Processi

Laboratorio Sistemi Operativi

Antonino Staiano

Email: antonino.staiano@uniparthenope.it

Processi

ESERCIZI

(14)

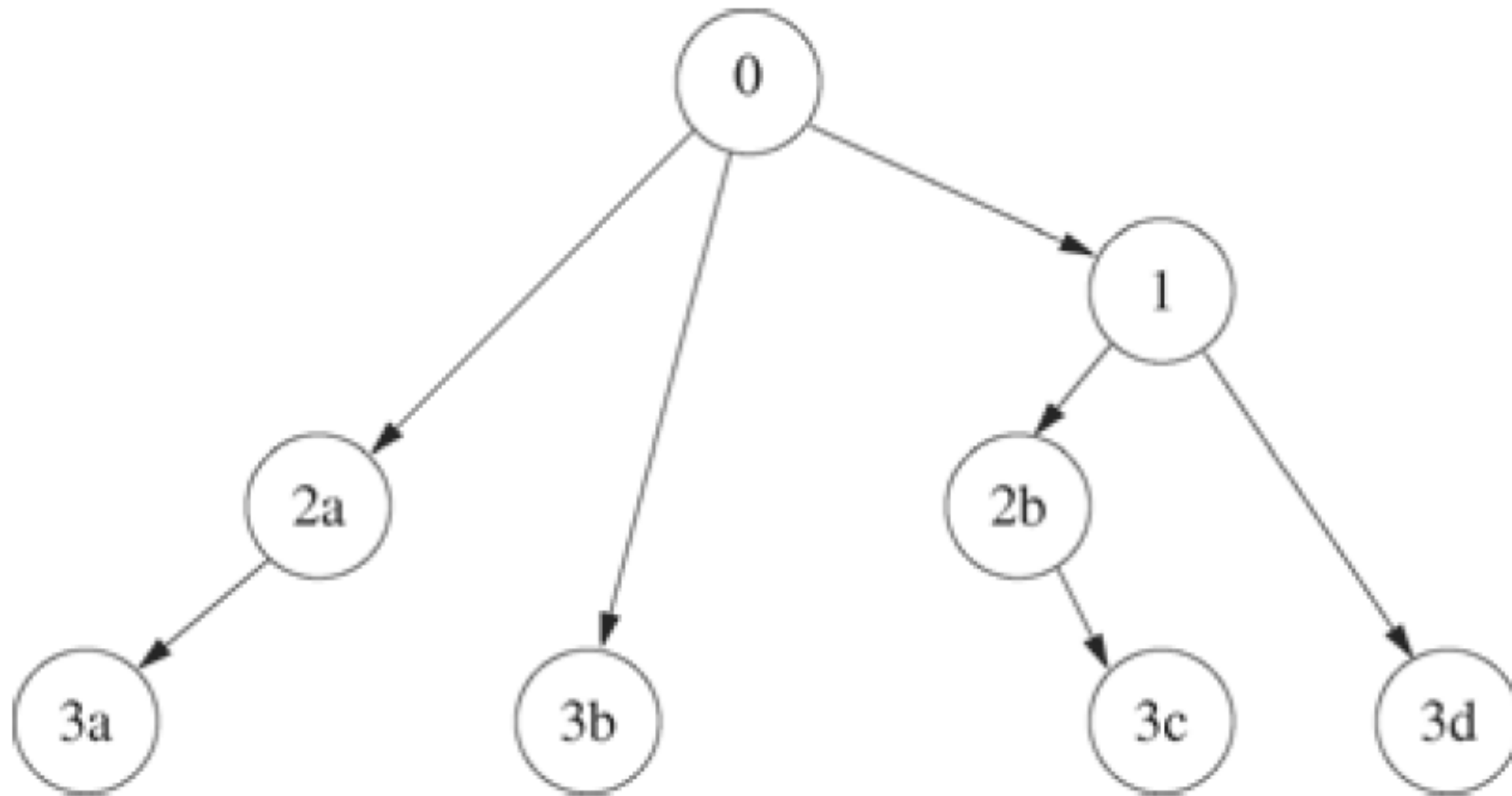
Esempio

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main (int argc, char *argv[]) {
    pid_t childpid = 0;
    int i, n;
    if (argc != 2){ /* Controllo argomenti*/
        fprintf(stderr, "Uso: %s processi\n", argv[0]);
        return 1;
    }
    n = atoi(argv[1]);
    for (i = 1; i < n; i++)
        if ((childpid = fork()) <= 0)
            break;
    fprintf(stderr, "i:%d processo ID:%d padre ID:%d figlio
    ID:%d\n",
    i, getpid(), getppid(), childpid);
    return 0;
}
```

Osservazioni

- Cosa succede se sostituiamo il test
 $(\text{childpid} = \text{fork}()) \leq 0$ con $(\text{childpid} = \text{fork}()) == -1$
- **Risposta:**
- Tutti i processi restano nel ciclo a meno che la fork fallisca. Ogni iterazione del ciclo raddoppia il numero di processi, formando un albero del tipo riportato in figura per $n=4$
 - In figura, ogni processo è rappresentato con un cerchio la cui etichetta è il valore di i al momento della sua creazione
 - Il processo originario ha etichetta 0
 - Le lettere distinguono processi creati con lo stesso valore di i
- In questo programma non si distingue tra padre e figlio dopo la fork
 - Entrambi procedono a creare figli alla successiva iterazione del ciclo

Esempio: albero dei processi creati ($n=4$)



Esercizio 1

```
int glob=5;
int pid=0;
pid=fork();
glob--;
pid=fork();
glob--;
if (pid!=0) {
    pid=fork();
    glob--;
}
printf("Valore di glob=%d\n",glob);
```

Esercizio 2

```
int glob=5;
int pid=0;
int main() {
    int i=0;
    for (i=1;i<3;i++) {
        pid=fork();
        if (pid==0) {
            glob=glob*2;
            sleep(i+1);
        }
        glob=glob+1;
        printf("Valore di glob=%d\n",glob);
    }
}
```

Esercizio 3

- Scrivere un programma C che:
 - Crea un processo figlio, stampa il messaggio “In attesa” ed attende la terminazione del figlio.
 - Il figlio esegue il comando “ls -l”
 - Quando il figlio termina, il padre visualizza il messaggio “nuovo figlio” e crea un secondo processo figlio.
 - Il secondo figlio aspetta per 5 secondi, stampa a video un messaggio e termina.
 - Quando il processo figlio termina, il padre stampa a video il pid del processo terminato.