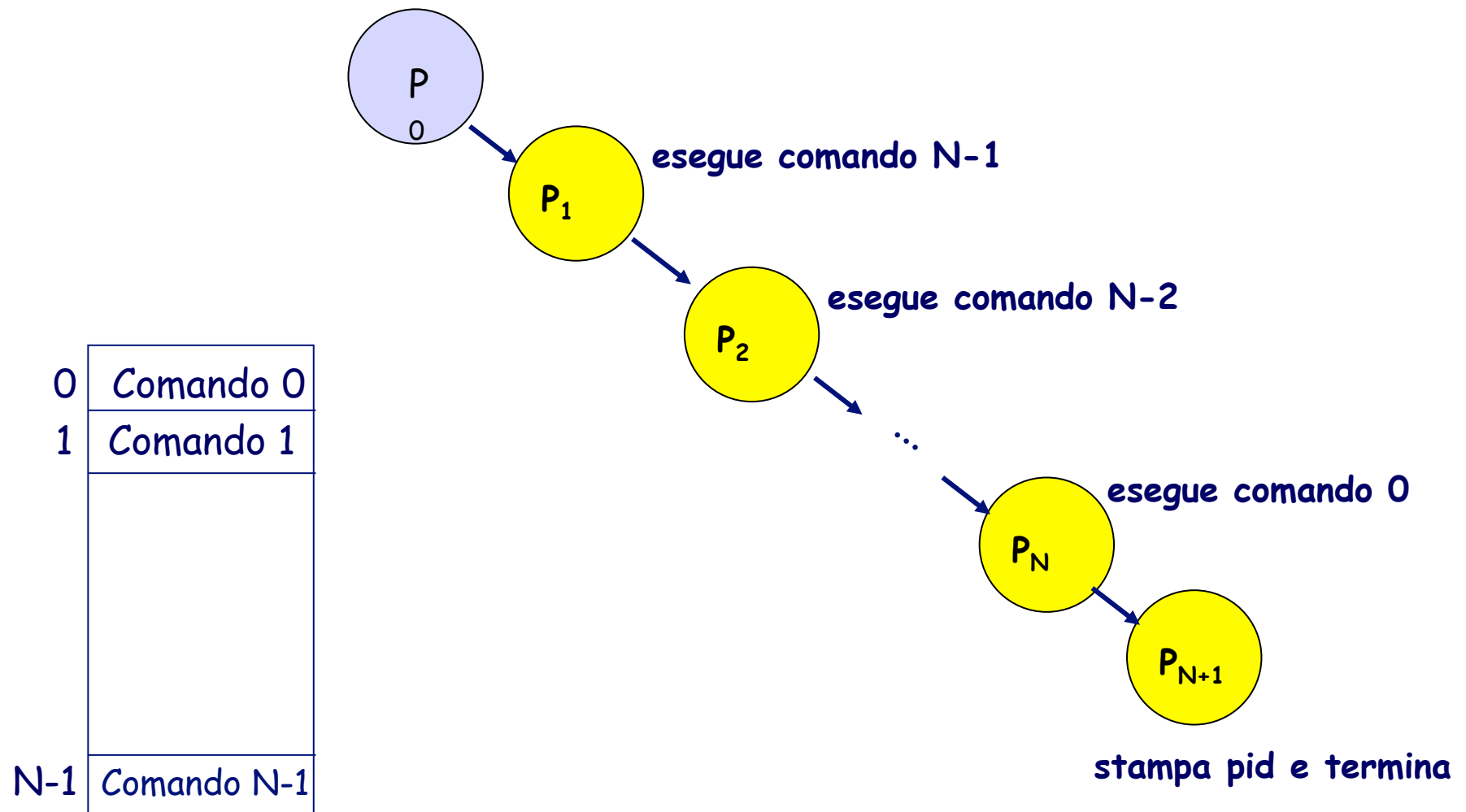


Esercizio sulla gestione di processi in Unix

Esercizio

Si vuole realizzare un programma C che, utilizzando le system call di Unix, rispetti le seguenti specifiche:

- il **processo iniziale** (P_0) deve acquisire da standard input una sequenza di N comandi (per semplicità, senza argomenti e senza opzioni), con N dato da input.
- Una volta letti gli N comandi, il processo iniziale P_0 deve dare origine ad una gerarchia lineare di processi di **profondità $N+1$** (figlio P_1 - nipote P_2 - bisnipote P_3 - ... - P_{N+1}): ogni processo della gerarchia dovrà eseguire un diverso comando della sequenza data; l'ultimo processo (P_{N+1}) della gerarchia, invece, dovrà semplicemente stampare il suo pid e terminare.



```

#include <stdio.h>
#define DIM 20
typedef char stringa[80];
typedef stringa strvett[DIM];
void gestoresequenza(int N, strvett vett);
int gest_stato(int S, int pid);

main()
{ int pid,ncom, stato, i;
  strvett vstr;
  printf("quanti comandi? ");
  scanf("%d", &ncom);
  for(i=0; i<ncom; i++)
  {      printf("\ndammi il prossimo comando(senza argomenti)");
        scanf("%s", vstr[i]);
  }
  gestoresequenza(ncom-1,vstr);
  pid=wait(&stato);
  gest_stato(stato,pid);
}

```

```

void gestoresequenza(int N, strvett vett)
{ int pid;
  pid=fork();
  if (pid==0) /* figlio*/
  {   if (N== -1) /* processo foglia */
      {   printf("\nfoglia %d: \n", getpid());
          exit(0);
        }
      else /*attivazione di un nuovo comando*/
      {   printf("\nProcesso %d per comando %s",getpid(),
                vett[N]);
          gestoresequenza(N-1, vett);
          execlp(vett[N], vett[N], (char *)0);
          perror("\nexec fallita: ");
          exit(-1);
        }
      }
}

```

```
int gest_stato(int S, int pid)
{
    printf("terminato processo figlio n.%d", pid);
    if ((char)S==0)
        printf("term. volontaria con stato %d", S>>8);
    else printf("terminazione involontaria per
segnale                %d\n", (char)S);
}
```

```
bash-2.05$ vi ese_proc.c
bash-2.05$ gcc -o eseproc ese_proc.c
bash-2.05$ eseproc
quanti comandi? 2
```

dammi il prossimo comando(senza argomenti)ps

dammi il prossimo comando(senza argomenti)who

Processo 5591 per comando who

Processo 5592 per comando ps

foglia 5593:

root	pts/2	May 6 17:29	(deis125.deis.unibo.it)
root	pts/3	May 6 17:29	(deis125.deis.unibo.it)
anna	pts/5	May 9 10:46	(deis136.deis.unibo.it)

PID	TTY	TIME	CMD
5542	pts/5	0:00	bash
5590	pts/5	0:00	eseproc
5592	pts/5	0:00	ps

terminato processo figlio n.5591

term. volontaria con stato 0

Spunti di discussione

- Output dei vari processi *mescolato: uso di file (o dispositivi) distinti*
- Come rilevare lo stato di terminazione di ogni processo???
- Pensare ad un'altra organizzazione per i processi:
 - ad esempio: gerarchia a 1 solo livello con N figli
- Realizzare una variante in cui i comandi siano dati come argomenti dalla linea di comando