POLITECNICO DI MILANO

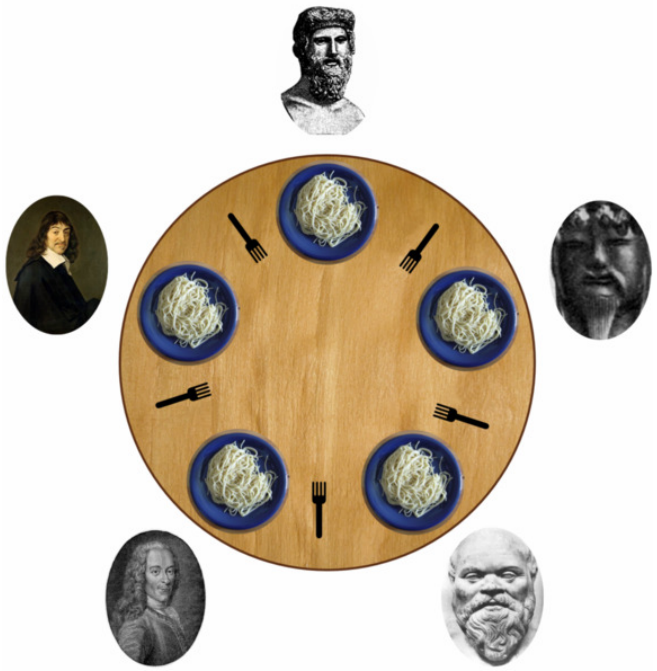
Esempio di tema d'esame

Laboratorio Software 2008-2009

M. Grotto

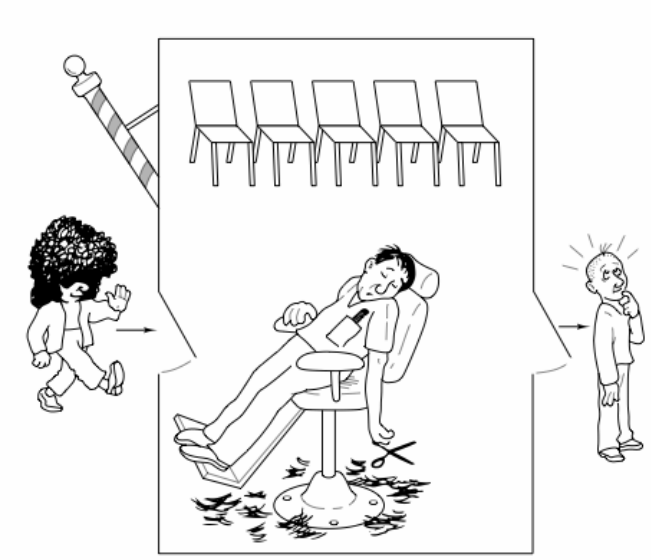
Esercizio sulla concorrenza

- Esiste una serie di problemi classici nella programmazione concorrente:
  - **Filosofi a cena:**
    formulato da Edsger Dijkstra come dining philosophers problem. Alcuni filosofi (5 nel testo originale) sono seduti a tavola di fronte al loro piatto ed a due forchette (condivise con i loro vicini). I filosofi alternano momenti durante i quali meditare e momenti durante i quali mangiare. Per mangiare devono prendere le due forchette accanto al loro piatto e mangiare mentre durante la meditazione devono tenere le forchette sul tavolo. Risulta evidente che il numero di forchette impedisce a tutti i filosofi di mangiare contemporaneamente quindi una corretta programmazione concorrente deve essere in grado di far mangiare alternativamente tutti i filosofi evitando che qualcuno in particolare soffra di starvation ed evitando che si verifichino stalli in fase di "acquisizione delle forchette".



Esercizio sulla concorrenza

- **Barbiere che dorme**:
  un barbiere possiede un negozio con una sola sedia da lavoro e un certo numero limitato di posti per attendere. Se non ci sono clienti il barbiere dorme altrimenti, all'arrivo del primo cliente il barbiere si sveglia ed inizia a servirlo. Se dovessero sopraggiungere clienti durante il periodo di attività del barbiere, essi si mettono in attesa sui posti disponibili. Al termine dei posti di attesa, un ulteriore cliente viene scartato. Questa problematica è molto vicina al sistema di funzionamento degli helpdesk informatizzati dove l'operatore serve, uno per volta, tutti i clienti in coda oppure attende, senza effettuare alcuna operazione in particolare, l'arrivo di nuove chiamate. Una corretta programmazione concorrente deve far "dormire" il barbiere in assenza di clienti, attivare il barbiere sul primo cliente al suo arrivo e mettere in coda tutti i successivi clienti tenendoli inattivi.



Codice da analizzare

- La comunicazione tra processi in Unix può essere fatta, per esempio, utilizzando le pipe (putki). Si spieghi cosa fa il seguente programma, indicando cosa viene prodotto a video.

Codice da analizzare

```
main()
{
  int putki[2], pid, N;
  if (pipe(putki) != 0) {perror("pipe"); exit(1);}
  pid = fork();
  if (pid == -1) {perror("fork"); exit(1);}
  if (pid != 0) {
      char message1[1024];
      close(putki[1]);
      if (read(putki[0],message1,1024) < 0)
         {perror("read putki"); exit(1);}
      printf("Child: %s\n",message1);
      close(putki[0]);
      printf("Parent: Yes.\n");
  }else{
      char message2[1024];
      close(putki[0]);
      strcpy(message2,"Can you hear me ?");
      N=strlen(message2);
      if (write(putki[1],message2,N) != N)
         {perror("write putki"); exit(2);}
      close(putki[1]);
  }
  exit(0);
}
```