1. Creazione e terminazione di processo

Scrivere un programma in cui, al momento dell'esecuzione, il processo padre crea un processo figlio e lo uccide (quale segnale si usa? vedere la documentazione di *signal* sul manuale). Predisporre cicli di attesa e stampe in modo da potere verificare la correttezza del programma.

2. Monitoraggio cambiamenti di stato dei processi figli

Scrivere un programma in cui un processo forka generando un figlio. Il genitore setta un handler per gestire i segnali relativi ai cambiamenti di stato del figlio.

Verificare che tale handler viene eseguito mediante invio di segnali al figlio da terminale.

Modificare l'handler ripristinando la disposizione di default, e verificarne l'effetto mediante invio di segnali al figlio da terminale.

Implementare tali passaggi due volte, sia con signal(), sia con sigaction().

3. Monitoraggio dell'esistenza di un processo

Scrivere un programma che prende in input un intero *P*. Al processo con pid *P* si invii un segnale *0*. All'interno del programma si testino due condizioni di errore: *ESRCH* (che significa? cercare nei lucidi della lezione) e *EPERM*. L'output del programma consiste in un messaggio che dice se il processo con *pid P* esiste o meno, e se abbiamo i diritti di inviargli un segnale.