```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

```
#include <sys/types.h>
#include <sys/wait.h>
pid_t wait(int *stat_loc);
pid_t waitpid(pid_t pid, int *stat_loc, int options);
```

```
#include <unistd.h>
char **environ;
int execl(const char *path, const char *arg0, ..., (char *)0);
int execlp(const char *file, const char *arg0, ..., (char *)0);
int execle(const char *path, const char *arg0, ..., (char *)0,
char *const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execve(const char *path, char *const argv[], char *const
envp[]);
```

```
#include <sys/types.h>
#include <signal.h>
int kill(pid_t pid, int sig);
```

```
#include <signal.h>
void (*signal(int sig, void (*func)(int)))(int);
```

```
#include <pthread.h>
int pthread_create(pthread_t *thread, pthread_attr_t *attr, void
*(*start_routine)(void *), void *arg);
```

```
void pthread_exit(void *retval);
int pthread_join(pthread_t th, void **thread_return);
int pthread_cancel(pthread_t thread);
int pthread_mutex_init(pthread_mutex_t *mutex, const
pthread_mutexattr_t *mutexattr);
int pthread_mutex_lock(pthread_mutex_t *mutex));
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

Utility:

- ```
  int rand (void);
  ```

  ```
  Per Es.:
        r=rand()%10; //per un numero compreso tra 0 e 10
  ```

- ```
  #include <unistd.h>
  unsigned int sleep(unsigned int seconds);
  ```