```
#include ...
int main(int argc, char* argv[])
{
    DIR *mydir;
    struct dirent *myfile;
    struct stat mystat;

    if (argc != 2)
        err_quit("usage: filepermission <directory_name>");

    if ((mydir = _____ == NULL)
        err_sys("can't open directory %s", argv[1]);

    printf("\n");

    char path[512];
    while((myfile = _____ != NULL)
    // while((myfile = readdir(mydir)))
    {
        sprintf(path, "%s/%s", argv[1], myfile->d_name);
        stat(path, &mystat);

        // file permissions
        mode_t val;
        val = (mystat.st_mode & ~S_IFMT);
        (val & S_IRUSR) ? printf(_____) : printf("-");
        (val & S_IWUSR) ? printf(_____) : printf("-");
        (val & S_IXUSR) ? printf(_____) : printf("-");
        (val & S_IRGRP) ? printf(_____) : printf("-");
        (val & S_IWGRP) ? printf(_____) : printf("-");
        (val & S_IXGRP) ? printf(_____) : printf("-");
        (_____) ? printf("r") : printf("-");
        (_____) ? printf("w") : printf("-");
        (_____) ? printf("x") : printf("-");

        // number of hard links
        printf("\t%d", _____.st_nlink);
        // user ID of owner
        printf("\t%d", _____.st_uid);
        // group ID of owner
        printf("\t%d"          .st_gid);
```

```
    // file permissions
    mode_t val;
    val = (mystat.st_mode & ~S_IFMT);
    (val & S_IRUSR) ? printf(_____) : printf("-");
    (val & S_IWUSR) ? printf(_____) : printf("-");
    (val & S_IXUSR) ? printf(_____) : printf("-");
    (val & S_IRGRP) ? printf(_____) : printf("-");
    (val & S_IWGRP) ? printf(_____) : printf("-");
    (val & S_IXGRP) ? printf(_____) : printf("-");
    (_____) ? printf("r") : printf("-");
    (_____) ? printf("w") : printf("-");
    (_____) ? printf("x") : printf("-");

    // number of hard links
    printf("\t%d", _____.st_nlink);
    // user ID of owner
    printf("\t%d", _____.st_uid);
    // group ID of owner
    printf("\t%d", _____.st_gid);
    // size in byte (for regular files)
    printf("\t%d", (int)_____.st_size);

    // last modification time
    struct tm *time_stamp = localtime(&_____.st_mtime);
    char buffer[80];

    strftime(buffer,10, "%b", time_stamp);

    printf("\t%4d %s %2d ", time_stamp->tm_year+1900, buffer, time_stamp->tm_mday);
    printf(" %s\n", myfile->d_name);
}

printf("\n");
closedir(mydir);
}
```

13. Descrivere nella riga sotto cosa fa il seguente programma

_____

13. Descrivere nella riga sotto cosa fa il seguente programma

_____

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc,char **argv) {
```



```
    pid_t pid;
    if(!(argc>1)) {
        fprintf(stderr,"Uso del programma errato \n");
        exit(1);
    }
    pid=fork();
    switch(pid) {
        case -1:
        perror("Errore nella chiamata a fork");
        exit(2);
        case 0:
        execvp(argv[1],&argv[1]);
        perror("Errore nella chiamata a execvp");
        exit(3);
        default:
        waitpid(pid,NULL,0);
        exit(0);
    }
}
```

14. Utilizzando esclusivamente comandi di shell bash, scrivere nella riga sotto cosa fa il seguente programma

```
/bin/ls -al / | /usr/bin/tr a-z A-Z
```

```
#include ...

void runpipe();

int main(int argc, char **argv) {
   int pid, status;
   int fd[2];

   pipe(fd);

   switch (pid = fork()) {
   case 0:
          runpipe(fd);
          exit(0);
   default:
          while ((pid = wait(&status)) != -1)
                 fprintf(stderr, "process %d exits with %d\n", pid, WEXITSTATUS(status));
          break;
   case -1:
          perror("fork");
          exit(1);
   }
   exit(0);
}

char *pippo[] = { "/usr/bin/tr", "a-z", "A-Z", 0 };
char *pluto[] = { "/bin/ls", "-al", "/", 0 };

void runpipe(int pfd[]) {
   int pid;

   switch (pid = fork()) {
   case 0:
          dup2(pfd[0], 0);
          close(pfd[1]);
          execvp(pippo[0], pippo);
          perror(pippo[0]);
   default:
          dup2(pfd[1], 1);
          close(pfd[0]);
          execvp(pluto[0], pluto);
          perror(pluto[0]);
   case -1:
          perror("fork");
          exit(1);
   }
}
```