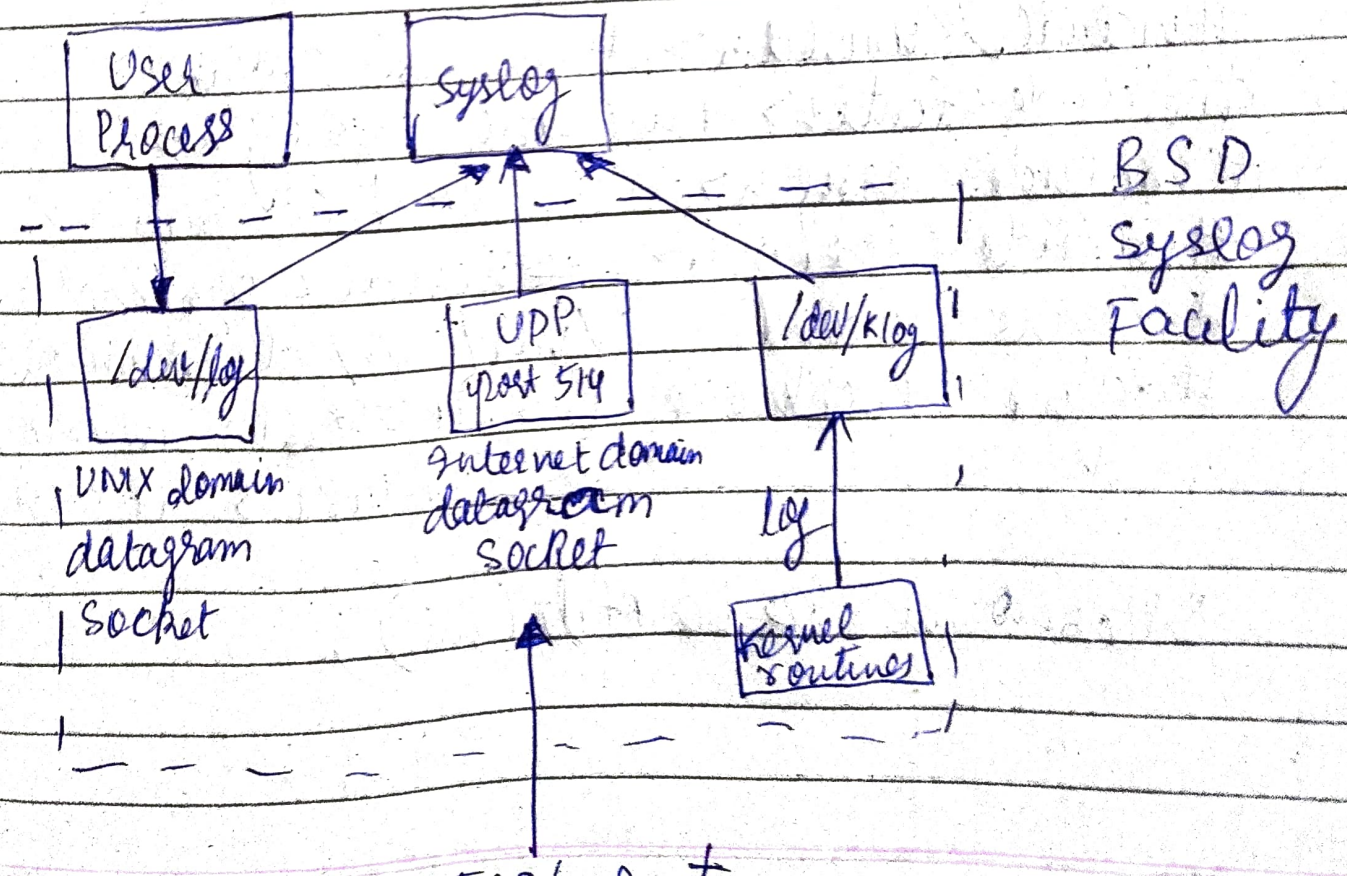
(3b) Design a C program to daemonize a process that logs their errors with the help of block diagram.

Ans: One Problem a daemon has is how to handle error messages. It can't simply write to standard error, since it should not have a controlling terminal. We don't want all the daemons writing to the console device, because on many workstations the console device runs a windowing system. A central daemon error logging facility is required. The BSD syslog facility was developed at Berkley & was used in 4.2BSD. The syslog function is included in XSI option in the Single UNIX Specification.



The interface to this facility is through the syslog function.

```
#include <syslog.h>
void openlog(const char *ident, int
             option, int facility);
void syslog(int priority, const char *
            format, ...);
void closelog(void);
int setlogmask(int maskpri);
                    return: previous log
                    priority mask
```

Ex: The func" below shows use of file & record locking to ensure only 1 copy of daemon is running

```
#include <unistd.h>          #include <stdlib.h>
#include <fcntl.h>           #include <syslog.h>
#include <string.h>          #include <errno.h>
#include <stdio.h>           #include <sys/stat.h>
#define lockfile "/var/run/daemon.pid"
#define lockmode (S_IRUSR|S_IWUSR|
                  S_IRGRP|S_IROTH)

extern int lockfile(int);
```

```
int already_running(void) {
int fd;
char buf[16];
fd = open(lockfile, O_RDWR|O_CREAT,
        LOCKMODE);
if (fd < 0) {
    syslog(LOG_ERR, "can't open %s: %s",
            lockfile, strerror(errno));
    exit(1);
}
if (lockfile(fd) < 0) {
    if (errno == EACCES || errno == EAGAIN) {
        close(fd);
        return 1;
    }
    syslog(LOG_ERR, "can't open %s: %s",
            lockfile, strerror(errno));
    exit(1);
}
ftruncate(fd, 0);
sprintf(buf, "%ld", (long) getpid());
write(fd, buf, strlen(buf)+1);
return 0;
}
```

⇒ Each copy of daemon tries to create a file & write its PID in the file. If file is already locked, the lockfile() function will fail with errno set to EACCES or EAGAIN so we return 1, indicating daemon is already running, else truncate file write PID & return 0.