Starting with Signals:-

For handling signals,

```c
#include "apue.h"

static void sig_usr(int);    /* one handler for both signals */

int
main(void)
{
    if (signal(SIGUSR1, sig_usr) == SIG_ERR)
        err_sys("can't catch SIGUSR1");
    if (signal(SIGUSR2, sig_usr) == SIG_ERR)
        err_sys("can't catch SIGUSR2");
    for ( ; ; )
        pause();
}

static void
sig_usr(int signo)          /* argument is signal number */
{
    if (signo == SIGUSR1)
        printf("received SIGUSR1\n");
    else if (signo == SIGUSR2)
        printf("received SIGUSR2\n");
    else
        err_dump("received signal %d\n", signo);
}
```

Also, a way to handle signals could be like:- In accordance with handling interrupt and quit signals,

```c
void sig_int(int), sig_quit(int);

if (signal(SIGINT, SIG_IGN) != SIG_IGN)
    signal(SIGINT, sig_int);
if (signal(SIGQUIT, SIG_IGN) != SIG_IGN)
    signal(SIGQUIT, sig_quit);
```

By this approach, the process catches the signal only if the signal is not currently being ignored.

And then we have unreliable signals:-
By this we mean that signals could get lost: a signal could occur and the process would never know about it. Also, a process had little control over a signal: a process could catch the signal or ignore it. Sometimes, we would like to tell the kernel to block a signal: don't ignore it, just remember if it occurs, and tell us later when we're ready.