```c
#ifndef socketError_
#define socketError_
#include <sys/socket.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

int Socket(int family, int type, int flags) {
  int sockfd;
  if ((sockfd = socket(family, type, flags)) < 0) {
    perror("socket");
    exit(EXIT_FAILURE);
  } else
    return sockfd;
}

void Bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen){
   if (bind(sockfd, addr, addrlen) == -1) {
      perror("bind");
      exit(EXIT_FAILURE);
   }
 }

void Listen(int sockfd, int backlog) {
  if (listen(sockfd, backlog) == -1) {
    perror("listen");
    exit(EXIT_FAILURE);
  }
}

int Accept(int listenfd, struct sockaddr *addr,socklen_t *addrlen) {
  int connfd;
  if ((connfd = accept(listenfd, addr, addrlen)) == -1 ) {
    perror("accept");
    exit(EXIT_FAILURE);
  }
  return connfd;
}

void Connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen){
   //conecte o socket com o endereco passado por argumento
   if (connect(sockfd, addr, addrlen) < 0) {
      perror("connect error");
      exit(EXIT_FAILURE);
   }
}

void Getsockname(int sockfd, struct sockaddr *addr, socklen_t *addrlen){
   //obtenha o endereco com o qual estamos comunicando
   if (getsockname(sockfd, addr, addrlen) < 0) {
     perror("getsockname error:");
     exit(EXIT_FAILURE);
   }
}

// leia count bytes de fd e ponha em buf
ssize_t Read(int fd, void *buf, size_t count) {
  ssize_t n = read(fd, buf, count);
  //reporte erros de read()
  if (n < 0) {
```



```c
    perror("read");
    exit(EXIT_FAILURE);
  }
  return n;
}

void Write(int fd, const void *buf, size_t count) {
  if (write(fd, buf, count) == -1) {
    perror("write");
    exit(EXIT_FAILURE);
  }
}

void Fputs(const char *s, FILE *stream) {
  if (fputs(s, stream) == EOF) {
    perror("fputs");
    exit(EXIT_FAILURE);
  }
}

void Inet_pton(int af, const char *src, void *dst) {
  if (inet_pton(af, src, dst) <= 0) {
    perror("inet_pton error");
    exit(EXIT_FAILURE);
  }
}

void Execv(const char *path, char *const argv[]) {
  execv(path, argv);
  perror("execvp"); // execve only returns on failure
  exit(EXIT_FAILURE);
}

void Pipe(int pipefd[2]) {
  if (pipe(pipefd) == -1) {
    perror("pipe");
    exit(EXIT_FAILURE);
  }
}

FILE* Fopen(const char *path, const char *mode) {
  FILE *f = fopen(path, mode);
  if (f == NULL) {
    perror("fopen");
    exit(EXIT_FAILURE);
  }
  return f;
}
#endif
```