Systems Programming Reference Card

```
int accept(int fd, struct sockaddr *addr, socklen_t *addrlen)
#include <sys/socket.h>
```

```
int bind(int fd, const struct sockaddr *addr, socklen_t addrlen)
#include <sys/socket.h>
```

```
int connect(int fd, const struct sockaddr *addr, socklen_t addrlen)
#include <sys/socket.h>
```

```
void freeaddrinfo(struct addrinfo *first)
#include <netdb.h>
#include <sys/socket.h>
```

```
void freeifaddrs(struct ifaddrs *first)
#include <ifaddrs.h>
#include <sys/types.h>
```

```
int getaddrinfo(const char *host, const char *service,
                const struct addrinfo *hints, struct addrinfo **first)
#define _POSIX_C_SOURCE >= 201112L
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
```

```
int gethostname(char *name, size_t length)
#define _GNU_SOURCE
#define _POSIX_C_SOURCE
#include <unistd.h>
```

```
int getifaddrs(struct ifaddrs **first)
#include <ifaddrs.h>
#include <sys/types.h>
```

```
int getnameinfo(const struct sockaddr *address, socklen_t addrlen,
                char *host, socklen_t hostlen, char* service,
                socklen servlen, int flags)
#define _GNU_SOURCE
#define _POSIX_C_SOURCE >= 201112L
#include <netdb.h>
#include <sys/socket.h>
```

```
uint16_t htons(uint16_t value)
#include <arpa/inet.h>
```

```
uint32_t htonl(uint32_t value)
#include <arpa/inet.h>
```

```
const char *inet_ntop(int domain, const void *net, char *pres, size_t length)
#include <arpa/inet.h>

int inet_pton(int domain, const char *pres, void *net)
#include <arpa/inet.h>

int listen(int fd, int backlog)
#include <sys/socket.h>

uint16_t ntohs(uint16_t value)
#include <arpa/inet.h>

uint32_t ntohl(uint32_t value)
#include <arpa/inet.h>

ssize_t recvfrom(int fd, void *buffer, size_t buflen, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen)
#include <sys/socket.h>

ssize_t sendto(int fd, const void *buffer, size_t buflen, int flags,
               const struct sockaddr *dest_addr, socklen_t *addrlen)
#include <sys/socket.h>

int setsockopt(int fd, int level, int optname, const void *optval,
               socklen_t optlen)
#include <sys/socket.h>

int socket(int domain, int type, int protocol)
#include <sys/socket.h>
```

```
// Generic
struct sockaddr {
 sa_family_t sa_family; // AF_UNIX, AF_INET, AF_INET6
 char sa_data[14];
};

// IPv4
struct in_addr {
 in_addr_t s_addr; // 32-bits
};

struct sockaddr_in {
 sa_family_t sin_family; // AF_INET
 in_port_t sin_port; // Port umber
 struct in_addr sin_addr;
 unsigned char __pad[X]; // Pad to size of struct sockaddr
};


struct ifaddrs {
  struct ifaddrs *ifa_next; // Next item in list
  char *ifa_name; // Name of interface
  unsigned int ifa_flags; // Flags from SIOCGIFFLAGS
  struct sockaddr *ifa_addr; // Address of interface
  struct sockaddr *ifa_netmask; // Netmask of interface
  union {
    struct sockaddr *ifu_broadaddr; // Broadcast address of interface
    struct sockaddr *ifu_dstaddr; // Point-to-point destination address
  } ifa_ifu;
  #define ifa_broadaddr ifa_ifu.ifu_broadaddr
  #define ifa_dstaddr ifa_ifu.ifu_dstaddr
  void *ifa_data; // Address-specific data
};


struct addrinfo {
  int ai_flags;
  int ai_family; // AF_INET, AF_INET6
  int ai_socktype; // SOCK_STREAM, SOCK_DGRAM
  int ai_protocol; // Protocol (name or number)
  socklen_t ai_addrlen; // sizeof(si_addr)
  struct sockaddr *ai_addr; // Socket address
  char *ai_canonname; // Canonical name of host (or NULL)
  struct addrinfo *ai_next; // Pointer to next struct in linked structure
};
```