CS311 - FA13: Final

Trevor Bramwell

December 10, 2013

1 Overview

This paper compares and contrasts Sockets, Anonymous Pipes, and Multiprocessing, between the Windows and POSIX APIs. Where parts are common, a direct API will comparison will be made between the function headers. For parts of the API that differ, an overview of the concepts between the two will be compared, along with Window's deviation from the POSIX specification.

2 Sockets

The first API I will be comparing is Sockets. In Windows these are referred to as *WinSock*. WinSock has the same commands for creating and accepting connections as POSIX sockets, with the addition of 'closesocket'.

2.1 socket

Windows: SOCKET WSAAPI socket(_In_ int af, _In_ int type, _In_ int protocol);

POSIX: int socket(int domain, int type, int protocol);

2.2 bind

Windows: int bind(_In_ SOCKET s, _In_ const struct sockaddr *name, _In_ int namelen);

POSIX: int bind(int socket, const struct sockaddr *address, socklen_t address_len);

2.3 listen

Windows: int listen(_In_ SOCKET s, _In_ int backlog);

POSIX: int listen(int socket, int backlog);

2.4 connect

Windows: int connect(_In_ SOCKET s, _In_ const struct sockaddr *name, _In_ int namelen);

POSIX: int connect(int socket, const struct sockaddr *address, socklen_t address_len);

2.5 accept

Windows: SOCKET accept(_In_ SOCKET s, _Out_ struct sockaddr *addr, _Inout_ int *addrlen);

POSIX: int accept(int socket, struct sockaddr *restrict address, socklen_t *restrict address_len);

2.6 shutdown

Windows: int shutdown(_In_ SOCKET s, _In_ int how);

POSIX: int shutdown(int socket, int how);

2.7 close/closesocket

Widows: int closesocket(_In_ SOCKET s);

POSIX: int close(int fildes);

2.8 send

Windows: int send(_In_ SOCKET s, _In_ const char *buf, _In_ int len, _In_ int flags);

POSIX: ssize_t send(int socket, const void *buffer, size_t length, int flags);

2.9 recv

Windows: int recv(_In_ SOCKET s, _Out_ char *buf, _In_ int len, _In_ int flags);

POSIX: ssize_t recv(int socket, void *buffer, size_t length, int flags);

2.10 sendto

Windows: int sendto(_In_ SOCKET s, _In_ const char *buf, _In_ int len, _In_ int flags, _In_ const struct sockaddr *to, _In_ int tolen);

POSIX: ssize_t sendto(int socket, const void *message, size_t length, int flags, const struct sockaddr *dest_addr, socklen_t dest_len);

2.11 recvfrom

Windows: int recvfrom(_In_ SOCKET s, _Out_ char *buf, _In_ int len, _In_ int flags, _Out_ struct sockaddr *from, _Inout_opt_ int *fromlen);

POSIX: ssize_t recvfrom(int socket, void *restrict buffer, size_t length, int flags, struct sockaddr *restrict address, socklen_t *restrict address_len);

3 Pipes and Named Pipes

4 Multiprocessing and Threads

5 References

5.1 Sockets

http://msdn.microsoft.com/en-us/library/windows/desktop/bb530741(v=vs.85).aspx

Windows API Reference: msdn.microsoft.com/en-us/library/windows/desktop/ms741394(v=vs.85).aspx POSIX References: http://pubs.opengroup.org/onlinepubs/9699919799/toc.htm

5.2 Pipes

http://msdn.microsoft.com/en-us/library/windows/desktop/aa365780(v=vs.85).aspx

http://msdn.microsoft.com/en-us/library/windows/desktop/aa365590(v=vs.85).aspx

5.3 Multiprocessing

http://msdn.microsoft.com/en-us/library/windows/desktop/ms684841(v=vs.85).aspx