Tytuł pracy: Interfejs programowy- gniazda BSD.

Autor: Marcin Mita IVFDS

STRESZCZENIE

Praca zawiera pokrótce podstawowe zagadnienia interfejsu gniazdowego.

Programy, które przeznaczone są dla sieci komputerowych korzystają z interfejsu dla programów użytkowych (ang. API – application program interface). Taki właśnie jest interfejs gniazdowy , zwany też gniazdami (ang. Sockets), a czasami gniazdami berkeleyowskimi lub gniazdami BSD (ang. Berkeley Software Distribution sockets). [4] Interfejs ten wywodzi się z systemu operacyjnego Unix, utworzonego w Barkeley.

Definicja gniazda: gniazda zaprojektowano jako interfejs między trzema górnymi warstwami modelu OSI a warstwa transportową. Trzy górne warstwy zajmują się wszystkimi szczegółami charakterystycznymi dla danego zastosowania sieci (np. FTP, Telnet czy HTTP) i niewiele wiedzą o szczegółach związanych z komunikacją. Dolne cztery warstwy niewiele wiedzą o rodzaju zastosowania, ale zajmują się wszystkimi zadaniami związanymi z komunikacją: wysyłanie danych, oczekiwaniem na potwierdzenie, szeregowaniem danych, które nie nadchodzą w sposób uporządkowany, obliczaniem i sprawdzaniem sum kontrolnych itd. Trzy górne warstwy tworzą tzw. proces użytkownika, podczas gdy cztery dolne warstwy zwykle stanowią część jądra systemu operacyjnego. System Unix umożliwia oddzielenie procesu użytkownika od jądra systemu, podobnie postępuje wiele innych współczesnych systemów operacyjnych, Dlatego naturalne miejsce, w którym powinno się zbudować interfejs programów użytkowych (API), znajduje się między czwartą a piątą warstwą modelu OSI. [1]

Następnie praca zwiera krótką historię oprogramowania sieciowego. Powstanie systemy operacyjnego BSD w Barkeley, inicjalizacja kolejnych odmian systemu.

Kolejno omówione zostały standardy systemu Unix takie jak: standard Posix, Stowarzyszenie Open Group powstałe w 1996 roku w wyniku połączenia stowarzyszenia X/Open Company (założonego w 1984 roku) i konsorcjum Open Software Foundation (OSF, założonego w 1988 roku), Stowarzyszenie IETF (ang. Internet Engineering Task Force).

Jako ostatni rozdział opisałem gniazdowe struktury adresowe dla dwóch protokołów:

- protokołu IPv4
- protokołu IPv6.

SPIS TREŚCI

1. WPROWADZENIE

1.1 Interfejs gniazdowy

Programy, które przeznaczone są dla sieci komputerowych korzystają z interfejsu dla programów użytkowych (ang. API – application program interface). Taki właśnie jest interfejs gniazdowy , zwany też gniazdami (ang. Sockets), a czasami gniazdami berkeleyowskimi lub gniazdami BSD (ang. Berkeley Software Distribution sockets). Interfejs ten wywodzi się z systemu operacyjnego Unix, utworzonego w Barkeley. [1]

Gniazda zaprojektowano jako interfejs między trzema górnymi warstwami modelu OSI a warstwa transportową. Trzy górne warstwy zajmują się wszystkimi szczegółami charakterystycznymi dla danego zastosowania sieci (np. FTP, Telnet czy HTTP) i niewiele wiedzą o szczegółach związanych z komunikacją. Dolne cztery warstwy niewiele wiedzą o rodzaju zastosowania, ale zajmują się wszystkimi zadaniami związanymi z komunikacją: wysyłanie danych, oczekiwaniem na potwierdzenie, szeregowaniem danych, które nie nadchodzą w sposób uporządkowany, obliczaniem i sprawdzaniem sum kontrolnych itd. Trzy górne warstwy tworzą tzw. proces użytkownika, podczas gdy cztery dolne warstwy zwykle stanowią część jądra systemu operacyjnego. System Unix umożliwia oddzielenie procesu użytkownika od jądra systemu, podobnie postępuje wiele innych współczesnych systemów operacyjnych, Dlatego naturalne miejsce, w którym powinno się zbudować interfejs programów użytkowych (API), znajduje się między czwartą a piątą warstwą modelu OSI. [1]

2. HISTORIA OPROGRAMOWANIA SIECIOWEGO W BSD

Interfejs gniazdowy wywodzi się z systemu operacyjnego 4.2BSD, utworzony w 1983 roku. W roku 1990 wprowadzono kilka zmian do interfejsu gniazdowego, ponieważ ukazało się wydanie systemu 4.3BSD Reno, w którym oprogramowanie protokołów OSI weszło do jądra systemu BSD. [2]

Wszystkie wydania systemu operacyjnego, począwszy od 4.2BSD i kończąc na 4.4BSD, powstały w Grupie Badawczej Systemów Komputerowych (ang. CSRG – Computer Systems Research Group) działającej na Uniwersytecie Kalifornijskim w Berkeley i wymagały uzyskania licencji dla kodu źródłowego systemu Unix. Ale całe oprogramowanie dotyczące pracy w sieci, zarówno należące do jądra systemu (takie jak stosy protokołów TCP/IP i dziedziny Unix, a także interfejs gniazd), jak i oprogramowanie użytkowe (takie jak programy klienta i serwera Telnet oraz klienta i serwera FTP) powstało niezależnie od kodu systemu Unix pochodzącego z firmy AT&T. Dlatego począwszy od roku 1989 można było uzyskać w Barkeley wydania systemu BSD zawierające pełny kod oprogramowania sieciowego, które wraz z różnymi innymi częściami systemu BSD nie było ograniczone licencją dla kodu źródłowego systemu Unix. Te wydania systemu uznano za „publicznie dostępne” i w końcu każdy mógł je otrzymać za pośrednictwem Internetu, korzystając z anonimowego FTP. [2]

Ostatnimi systemami utworzonymi w Berkeley były: 4.3BSD-Lite z 1994 roku oraz 4.4BSD-Lite2 z 1995 roku. Zauważmy, że tych dwóch systemów użyto następnie jako podstawy innych systemów, wśród których cztery: BSD/OS, FreeBSD, NetBSD i OpenBSD są nadal rozwijane i rozszerzane. [2]

Punktem wyjścia dla wielu wersji systemu Unix był kod oprogramowania sieciowego jakiejś wersji systemu BSD, zawierającego oprogramowanie interfejsu gniazdowego; tak utworzone systemy nazywamy implementacjami berkeleyowskimi (ang. Berkeley – derived implementation). Wiele odpłatnych wersji systemu Unix oparto na Wydaniu 4 Systemu V (ang. SVR4 – System V Release 4); niektóre z nich mają kod oprogramowania sieciowego wywodzący się z systemu Berkeley (np. UnixWare 2.x), podczas gdy kod oprogramowania sieciowego w innych systemach SVR4 utworzono niezależnie (np. Solaris 2x). Zauważmy, że

system Linux, który jest popularną, darmową implementacją systemu Unix, nie jest implementacją berkeleyowską, ponieważ jego oprogramowanie sieciowe i interfejs gniazdowy utworzono całkowicie od podstaw, bez korzystania z gotowego kodu. [2]

3. STANDARDY SYSTEMU UNIX

3.1 Posix

Nazwa Posix jest akronimem oznaczającym Przenośny Interfejs Systemu Operacyjnego (ang. Portable Operating System Interface) i odnosi się nie do jednego standardu, lecz do rodziny standardów opracowanych w Instytucie Inżynierów Elektryczności i Elektroniki – IEEE (ang. Institute for Electrical and Electronics Engineers, Inc.). Standardy należące do tej rodziny uznano za normy Międzynarodowej Organizacji Normalizacyjnej (ISO) oraz Międzynarodowej Komisji Elektrotechnicznej – IEC (ang. International Electrotechnical Commission), w skrócie nazywanej ISO/IEC. [3]

Pierwszy standard z tej rodziny nazywał się IEEE Std 10003.1-1988 (317 stron) i definiował interfejs między językiem C, a jądrem systemu podobnego do systemu Unix. Obejmował on: podstawowe narzędzia pozwalające tworzyć oprogramowanie wieloprocesowe (funkcje fork i exec, sygnały, liczniki zegarowe), środowisko działania procesu (identyfikatory użytkowników, grupy procesów), pliki i katalogi (wszystkie funkcje wejścia-wyjścia), wejście-wyjście terminali, bazy danych systemowych (plik haseł i plik grup) oraz formaty archiwalne tar oraz cpio. [3]

Pierwszym standardem Posix była wersja próbna standardu, wprowadzona w 1986 roku i znana pod nazwą IEEEIX. Nazwę Posix zaproponował Richard Stallman. [3]

Ten stendard uaktualniono w roku 1990 i opublikowano jako IEEE Std 1003.1-1990 (356 stron); była to również międzynarodowa norma ISO/IEC 9945-1: 1990. Wersja z 1990 roku różniła się minimalnie od wersji z 1988 roku. Tytuł jej uzupełniono o frazę: „Część 1: Systemowy Interfejs Programów Użytkowych (API) [język C]”, wskazując w ten sposób, że jest to interfejs dla języka C. [3]

Następny standard Posix nazywał się IEEE Std 1003.2-1992 i miał podtytuł: „Część 2: Shell i usługi”. Opublikowano go w dwóch tomach, zajmujących ogółem około 1300 stron. Zawierał definicję shella (opartą na shellu Bourne’a Systemu V) oraz około 100 programów usługowych (programów zazwyczaj wykonywanych z poziomu shella, poczynając od awk oraz basename, kończąc na vi oraz yacc). [3]

Następnie powstał standard IEEE Std 1003.1b-1993. Wywodził się on ze standardu IEEE P1003.4, który uaktualniono, uzupełniając go o opracowane przez grupę roboczą P1003.4 rozszerzenia dotyczące czasu rzeczywistego, tworząc w ten sposób standard 1003.1-1990. Później dodano do niego synchronizację plików, wejście-wyjście asynchroniczne, semafory, zarządzanie pamięcią (funkcję mmap i pamięć wspólną), szeregowanie zadań, zegary i liczniki zegarowe oraz obsługę kolejek komunikatów. Standard IEEE Std 1003.1b-1993 zajmuje 590 stron. [3]

Potem powstał standard IEEE Std 1003.1, wydanie 1996 [40], który zawiera standardy: IEEE Std 1003.1-1990 (podstawowy interfejs programów użytkowych), IEEE Std 1003.1b-1993 (rozszerzenie dotyczące czasu rzeczywistego), IEEE Std 1003.1c-1995 (obsługa wątków) oraz IEEE Std 1003.1i-1995 (techniczne poprawki wprowadzone do standardu IEEE Std 1003.1b). Po dodaniu trzech rozdziałów o wątkach standard ten, który nazywał się również normą ISO/IEC 9945-1: 1996, zajmuje 743 strony. [3]

Standard Posix.1 składa się z następujących części:

- Część 1: Systemowy interfejs programów użytkowych (API) [język C]

- Część 2: Shell i programy usługowe
- Część 3: Administracja systemu (w opracowaniu)

Standardem Posix jest IEEE Std 1003.1g, pod tytułem: „Interfejsy niezależne od protokołu”, zwany PII (ang. Protocol Independent Interfaces), utworzony przez grupę roboczą P1003.1g. Jest to standardowy interfejs programów użytkowych wykonywanych w sieci komputerowej i zawiera dwa szczegółowe interfejsy, zwane DNI (ang. Detailed Network Interface):

1. DNI/Socket, oparty na interfejsie gniazd 4.4BSD
2. DNI/XTI, oparty na definicji X/Open XPG4 [3]

3.2 Stowarzyszenie Open Group

Stowarzyszenie Open Group powstało w 1996 roku w wyniku połączenia stowarzyszenia X/Open Company (założonego w 1984 roku) i konsorcjum Open Software Foundation (OSF, założonego w 1988 roku). Open Group jest międzynarodowym stowarzyszeniem zrzeszającym producentów i klientów wywodzących się ze świata przemysłu, administracji państwowej i nauki. [3]

Stowarzyszenie X/Open wydało w 1989 roku publikację pt. „X/Open Portability Guide”, wyd. 3 (XPG3). Wydanie 4 tej publikacji ukazało się w 1992 roku, a w 1994 roku ukazała się wersja 2 wydania 4. Ta ostatnia wersja była znana również pod nazwą Spec 1170, przy czym magiczna liczba 1170 była sumą liczb interfejsów systemowych (926), plików nagłówkowych (70) i poleceń (174). Ostatnio ten zbiór definicji otrzymał nazwę X/Open Single Unix Specification, ale często bywa nazywany Unix 95. [3]

3.3 Stowarzyszenie IETF

Stowarzyszenie IETF (ang. Internet Engineering Task Force) jest wielkim otwartym międzynarodowym zgromadzeniem projektantów sieci komputerowych, operatorów, producentów oraz naukowców zainteresowanych rozwojem architektury Internetu i wygodnym korzystaniem z jego usług. Do stowarzyszenia IETF może należeć każda osoba przejawiająca podobne zainteresowania. [3]

Powstawanie standardów Internetu jest udokumentowane w publikacji RFC 2026 [15]. Standardy Internetu dotyczą zazwyczaj protokołów, nie zaś interfejsów programów użytkowych. Niemniej jednak dwa dokumenty RFC, [32] i [96], określają interfejs gniazdowy dla wersji 6 protokołu IP. Nie są to definicje standardów, lecz dokumenty informacyjne; opublikowano je w celu przyspieszenia prac nad przenośnym oprogramowaniem użytkowym, prowadzonych przez wielu producentów, którzy pracowali nad wcześniejszymi wersjami IPv6. Opracowywanie i zatwierdzanie standardów trwa zazwyczaj dość długo. Niemniej jednak po pewnym czasie standardy interfejsów dla protokołu IPv6 będą prawdopodobnie uznane w sposób bardziej formalny. [3]

4. GNIAZDA

Programy, które przeznaczone są dla sieci komputerowych korzystają z interfejsu dla programów użytkowych (ang. API – application program interface). Taki właśnie jest interfejs gniazdowy , zwany też gniazdami (ang. Sockets), a czasami gniazdami berkeleyowskimi lub gniazdami BSD (ang. Berkeley Software Distribution sockets). Interfejs ten wywodzi się z systemu operacyjnego Unix, utworzonego w Barkeley. [4]

W tym punkcie wymienię podstawowe funkcje, które dokonują przekształceń między tekstową reprezentacją adresu, a jego wartością w postaci liczby dwójkowej, którą umieszczają w gniazdowej strukturze adresowej.

4.1 Gniazdowe struktury adresowe

W większości funkcji gniazd trzeba jako argument podać wskaźnik do gniazdowej struktury adresowej. Dla każdej rodziny protokołów zdefiniowano właściwą jej gniazdową strukturę adresową. Nazwy tych struktur zaczynają się od członu sockaddr_, po którym występuje drugi człon nazwy, identyfikujący daną rodzinę protokołów. [4]

4.2 Gniazdowe struktura adresowa dla protokołu IPv4

Gniazdowa struktura adresowa dla protokołu IPv4, zwana często internetową gniazdową strukturą adresową, nazywa się sockaddr_in; definiuję się ją, dołączając plik nagłówkowy <netinet/in.h>. [4]

Równocześnie z powstaniem systemu operacyjnego 4.3BSD-Reno, gdy utworzono oprogramowanie dla protokołów OSI, wtedy do struktury dodano pierwszą składową określającą długość, sin_len. Przed ukazaniem się tej wersji systemu BSD jako pierwsza składowa struktury występowało pole sin_family, które tradycyjnie miało typ unsigned short. Nie wszyscy producenci uwzględnili pole długości w gniazdowej strukturze adresowej i standerd Posix.1g nie wymaga określenia tej składowej. [4]

Określenie pola długości upraszcza obsługę struktur mających zmienną długość. Jeżeli nawet pole długości występuje w strukturze, to nigdy nie musimy go zapełniać ani sprawdzać jego wartości, chyba że mamy do czynienia z gniazdami wyznaczania tras. Z pól tych korzystają procedury znajdujące się w jądrze systemu, które zajmują się gniazdowymi strukturami adresowymi dla różnych rodzin protokołów (np. oprogramowanie tabeli tras). [4]

4.3 Gniazdowe struktura adresowa dla protokołu IPv6

Adres gniazda dla protokołu IPv6 jest zdefiniowany w wyniku dołączenia pliku nagłówkowego <netinet/in.h> do programu. [4]

Gniazdowa struktura musi zawierać następujące kwestie:

- Zdefiniowaną stała SIN6_LEN, jeżeli system obsługuje składową określającą długość gniazdowej struktury adresowej.
- Rodzinie protokołów IPv6 odpowiada stała AF_INET6, podczas gdy rodzinie IPv4 – stała AF_INET.
- Składowe tej struktury są tak uporządkowane, że jeżeli struktura socksddr_in6 jest umieszczona w pamięci z uwzględnieniem wyrównania 64-bitowego, to tak samo wyrównana jest również jej 128-bitowa składowa sin6_addr. W niektórych procesorach 64-bitowych dostęp do 64-bitowych wartości jest zoptymalizowany wówczas, gdy są one umieszczane w pamięci co 64 bity.
- Składowa sin6_flowinfo jest podzielona na trzy pola:
  - etykieta przepływu znajduje się w 24 najmniej znaczących bitach,
  - następne 4 bity zajmuje priorytet,
  - następne 4 bity są zarezerwowane [4]

4.4 Podsumowanie gniazdowych struktur adresowych

Gniazdowe struktury adresowe są integralną częścią każdego programu przeznaczonego dla sieci komputerowych. Przydzielamy im miejsce w pamięci, zapełniamy je odpowiednią zawartością i różnym funkcjom gniazd przekazujemy wskaźniki do tych struktur. Czasami wskaźnik do jednej z tych struktur przekazujemy do takiej funkcji gniazd, która zapełnia tę strukturę jej zawartością. Struktury przekazujemy zawsze przez odniesienie (tzn. przekazujemy wskaźnik do struktury, nie zaś samą strukturę) i zawsze przekazujemy rozmiar struktury jako odrębny argument funkcji. Kiedy funkcja gniazd zapełnia strukturę, wtedy jej długość też przekazujemy przez odniesienie, aby funkcja mogła ją uaktualnić; tego rodzaju argument funkcji nazywamy argumentem-wynikiem. [4]

Gniazdowe struktury adresowe są samo się definiujące, ponieważ zawsze zaczynają się od pola identyfikującego rodzinę adresów zawartych w danej strukturze. W nowszych implementacjach, w których gniazdowe struktury adresowe mogą mieć zmienną długość, struktury te zawierają na początku również pole określające długość całej struktury.

Dwie funkcje, które adresy IP z postaci prezentacji (której używamy, zapisując adresy za pomocą znaków w kodzie ASCII) przekształcają do postaci liczbowej (w której umieszcza się je w strukturze) i z powrotem, nazywają się inet_pton oraz inet_ntop. Funkcje te są zależne od protokołu. [4]

Gniazda TCP dostarczają dane do programu użytkowego jako strumień bajtów; w przesyłanych danych nie używa się znaczników rekordów. Wartość przekazywana przez funkcję read może być mniejsza niż ta, o którą prosiliśmy, ale nie spowoduje to zasygnalizowania błędu. [4]

LITERATURA

[1],[2],[3],[4]: W. Richard Stevens „Unix programowanie usług sieciowych“ cz. 1 (API: gniazda i XTI) Wydawnictwo Naukowo-Techniczne Warszawa 2000 r.