

Programowanie usług sieciowych w systemie *Unix*

Spis funkcji interfejsu gniazd

Przedstawimy teraz najważniejsze funkcje interfejsu gniazd systemu Unix.

1 Funkcja `socket`

```
int socket(int family, int type, int protocol);
```

Funkcja tworzy gniazdo dla żądanego protokołu. Używana zarówno z oprogramowaniu klienta jak i serwera.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `family` – oznacza rodzinę adresów, można przekazać jedną ze stałych: `AF_INET` – protokół IPv4, `AF_INET6` – protokół IPv6 i inne;
- `type` – rodzaj gniazda, można przekazać jedną ze stałych `SOCK_STREAM` dla gniazda strumieniowego (TCP), `SOCK_DGRAM` dla gniazda datagramowego (UDP) oraz `SOCK_RAW` dla gniazda surowego (IP);
- `protocol` – przekazujemy 0 z wyjątkiem gniazd surowych, w których ten argument ma znaczenie.

Wartości zwracane:

- deskryptor gniazda (nieujemna liczba całkowita przydzielona przez system dla gniazda),
- -1, jeśli wystąpił błąd.

2 Funkcja `connect`

```
int connect(int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);
```

Nawiązuje połączenie (aktywnie) w przypadku gniazda TCP, łączy gniazdo z adresem w jądrze systemu w przypadku UDP. Funkcja ta jest używana w oprogramowaniu klienta.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `sockfd` – deskryptor gniazda, z którego ma być nawiązane połączenie (gniazdo utworzone wcześniej za pomocą funkcji `socket`);
- `servaddr` – gniazdowa struktura adresowa zawierająca adres i port serwera, z którym nawiązane ma być połączenie;
- `addrlen` – długość (wielkość) struktury `servaddr` (w bajtach).

Wartości zwracane:

- 0, jeśli wszystko w porządku,
- -1, jeśli wystąpił błąd.

3 Funkcja bind

```
int bind(int sockfd, const struct sockaddr *myaddr, socklen_t addrlen);
```

Funkcja "spina" gniazdo z adresem lokalnym w jądrze systemu. Należy ją wywołać po utworzeniu gniazda (za pomocą funkcji `socket`) a przed rozpoczęciem nasłuchiwanie (funkcja `listen`). Funkcja ta używana jest zazwyczaj w oprogramowaniu serwera, choć wyjątkowo może być również użyta w programie klienta.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `sockfd` – deskryptor gniazda, które ma być połączone z adresem lokalnym (gniazdo utworzone wcześniej za pomocą funkcji `socket`);
- `myaddr` – gniazdowa struktura adresowa zawierająca adres i port lokalny, z którym ma być powiązane gniazdo (aby nasłuchiwać na wszystkich interfejsach sieciowych używa się w tej strukturze zamiast adresu stałej `INADDR_ANY` przekształconej przez funkcję `htonl`);
- `addrlen` – długość (wielkość) struktury `myaddr` (w bajtach).

Wartości zwracane:

- 0, jeśli wszystko w porządku,
- -1, jeśli wystąpił błąd.

4 Funkcja listen

```
int listen(int sockfd, int backlog);
```

Funkcja rozpoczyna nasłuchiwanie na gnieździe podanym jako parametr. Gniazdo to musi być wcześniej skojarzone ze strukturą adresową za pomocą funkcji `bind`. Funkcja ta jest używana w oprogramowaniu serwera.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `sockfd` – deskryptor gniazda, na którym ma się rozpocząć nasłuchiwanie;
- `backlog` – maksymalna sumaryczna długość kolejek dla połączeń nawiązywanych i nawiązanych, ale jeszcze nie obsłużonych.

Wartości zwracane:

- 0, jeśli wszystko w porządku,
- -1, jeśli wystąpił błąd.

5 Funkcja accept

```
int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);
```

Jest to funkcja blokująca używana w oprogramowaniu serwerowym. Oczekuje ona na nawiązanie połączenia przez klienta, kończy działania, gdy połączenie zostanie nawiązane. Wywołuje się ją zazwyczaj w pętli, po wywołaniu funkcji `listen`.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `sockfd` – deskryptor gniazda nasłuchującego, na którym oczekujemy na połączenie;
- `cliaddr` – wskaźnik do struktury adresowej, w której funkcja umieścić ma dane adresowe klienta, jeśli ich nie potrzebujemy przekazujemy `NULL`;
- `addrlen` – wskaźnik do liczby całkowitej, w której funkcja umieścić ma długość struktury będącej argumentem `cliaddr`, jeśli `cliaddr` ma wartość `NULL` ten parametr powinien również mieć wartość `NULL`.

Wartości zwracane:

- nieujemny deskryptor, jeśli wszystko w porządku,
- -1, jeśli wystąpił błąd.

6 Funkcja `read`

```
ssize_t read(int fd, void *buff, size_t bufflen);;
```

Jest to funkcja blokująca służąca do odczytywania danych z podanego deskryptora, który może być deskryptorem pliku lub gniazda sieciowego.

Plik nagłówkowy: `unistd.h`

Argumenty:

- `fd` – deskryptor pliku lub gniazda z którego chcemy czytać;
- `buff` – wskaźnik do miejsca w pamięci, w którym mają być umieszczone odczytane dane (zazwyczaj jest to tablica znaków);
- `bufflen` – długość (w bajtach) miejsca przeznaczonego na odczytane dane.

Wartości zwracane:

- ilość odczytanych bajtów (nieujemna), jeśli odczyt się powiódł,
- -1, jeśli wystąpił błąd.

Uwagi: Stosowana do gniazd połączonych, zazwyczaj TCP.

7 Funkcja `write`

```
ssize_t write(int fd, void *buff, size_t bufflen);;
```

Jest to funkcja (blokująca) służąca do wysyłania danych (pisania) do podanego deskryptora, który może być deskryptorem pliku lub gniazda sieciowego.

Plik nagłówkowy: `unistd.h`

Argumenty:

- `fd` – deskryptor pliku lub gniazda do którego chcemy pisać;
- `buff` – wskaźnik do miejsca w pamięci, w którym znajdują się dane do wypisania;
- `bufflen` – ilość danych (w bajtach), które mają zostać wypisane.

Wartości zwracane:

- ilość faktycznie wysłanych (wypisanych) bajtów (nieujemna i mniejsza lub równa od argumentu `bufflen`), jeśli pisanie się powiodło,
- -1, jeśli wystąpił błąd.

Uwagi: Stosowana do gniazd połączonych, zazwyczaj TCP.

8 Funkcja `recvfrom`

```
ssize_t recvfrom(int sockfd, void *buff, size_t buflen, int flags,  
                 struct sockaddr *from, socklen_t *fromlen);
```

Jest to funkcja blokująca służąca do odczytu z gniazda połączanego lub nie.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `sockfd` – deskryptor gniazda, z którego chcemy czytać;
- `buff` – wskaźnik do miejsca w pamięci, w którym mają być umieszczone odczytane dane (zazwyczaj jest to tablica znaków);
- `buflen` – długość (w bajtach) miejsca przeznaczonego na odczytane dane;
- `flags` – flagi, czyli dodatkowe opcje odczytu (dla standardowego odczytu przekazujemy 0);
- `from` – wskaźnik do struktury adresowej, w której funkcja umieścić ma dane adresowe nadawcy, jeśli ich nie potrzebujemy przekazujemy `NULL`;
- `fromlen` – wskaźnik do liczby całkowitej, w której funkcja umieścić ma długość struktury będącej argumentem `from`, jeśli `from` ma wartość `NULL` ten parametr powinien również mieć wartość `NULL`.

Wartości zwracane:

- ilość odczytanych bajtów (nieujemna), jeśli odczyt się powiódł,
- -1, jeśli wystąpił błąd.

Uwagi: Może być stosowana do gniazd niepołączonych, zazwyczaj UDP.

9 Funkcja `sendto`

```
ssize_t sendto(int sockfd, const void *buff, size_t buflen, int flags,  
               const struct sockaddr *to, socklen_t tolen);
```

Jest to funkcja (blokująca) służąca do wysłania danych do gniazda połączanego lub nie.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `sockfd` – deskryptor gniazda, do którego chcemy pisać;
- `buff` – wskaźnik do miejsca w pamięci, w którym są umieszczone dane do wysłania;
- `buflen` – ilość danych (w bajtach), które chcemy wysłać;
- `flags` – flagi, czyli dodatkowe opcje odczytu (dla standardowego odczytu przekazujemy 0);
- `to` – wskaźnik do struktury adresowej, zawierającej adres odbiorcy, jeśli gniazdo jest połączone przekazujemy `NULL`;
- `tolen` – długość struktury będącej argumentem `to`, jeśli `to` ma wartość `NULL` ten parametr powinien również mieć wartość `NULL`.

Wartości zwracane:

- ilość faktycznie wysłanych (wypisanych) bajtów (nieujemna i mniejsza lub równa od argumentu `buflen`), jeśli pisanie się powiodło,
- -1, jeśli wystąpił błąd.

Uwagi: Może być stosowana do gniazd niepołączonych, zazwyczaj UDP.

10 Funkcja `fork`

```
pid_t fork(void);
```

Funkcja powoduje utworzenie nowego procesu będącego kopią procesu rodzica (tego, który wywołał funkcję `fork`).

Plik nagłówkowy: `unistd.h`

Argumenty: brak.

Wartości zwracane:

- identyfikator procesu potomnego zwracany jest do procesu rodzica (nie ma innej możliwości uzyskania przez rodzica identyfikatora procesu potomnego),
- 0 zwracane jest do procesu potomnego (identyfikator procesu rodzica może być uzyskany za pomocą funkcji `getppid`).

Uwagi: Jedno wywołanie funkcji powoduje dwa z niej powroty (w dwóch różnych procesach).

11 Funkcja `select`

```
int select(int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *exceptset,  
           const struct timeval *timeout);
```

Jest to funkcja blokująca, która oczekuje, aż jeden z deskryptorów będzie gotowy do czytania, pisania lub pobrania wyjątku.

Plik nagłówkowy: `sys/socket.h`

Argumenty:

- `maxfdp1` – liczba o 1 większa od największego z przekazanych deskryptorów;
- `readset` – zbiór deskryptorów, dla których oczekujemy na możliwość czytania, jeśli takich nie ma przekazujemy `NULL`;
- `writeset` – zbiór deskryptorów, dla których oczekujemy na możliwość pisania, jeśli takich nie ma przekazujemy `NULL`;
- `exceptset` – zbiór deskryptorów, dla których oczekujemy na wystąpienie wyjątku, jeśli takich nie ma przekazujemy `NULL`;
- `timeout` – maksymalny czas oczekiwania (w sekundach i mikrosekundach), jeśli chcemy czekać dowolnie długo przekazujemy `NULL`.

Wartości zwracane:

- łączna ilość deskryptorów (spośród przekazanych do funkcji) gotowych do czytania, pisania i pobrania wyjątków,
- 0, jeśli upłynął zadany czas, a żaden z podanych deskryptorów nie jest gotowy,
- -1, jeśli wystąpił błąd.

Uwagi: Do operowania na typie `fd_set` służą następujące makrodefinicje:

- `void FD_ZERO(fd_set *fdset);` – zeruje zbiór,
- `void FD_SET(int fd, fd_set *fdset);` – dodaje do zbioru deskryptor `fd`,
- `void FD_CLR(int fd, fd_set *fdset);` – usuwa ze zbioru deskryptor `fd`,
- `int FD_ISSET(int fd, fd_set *fdset);` – zwraca prawdę (1) jeśli deskryptor `fd` jest w zbiorze, fałsz (0), jeśli go nie ma.