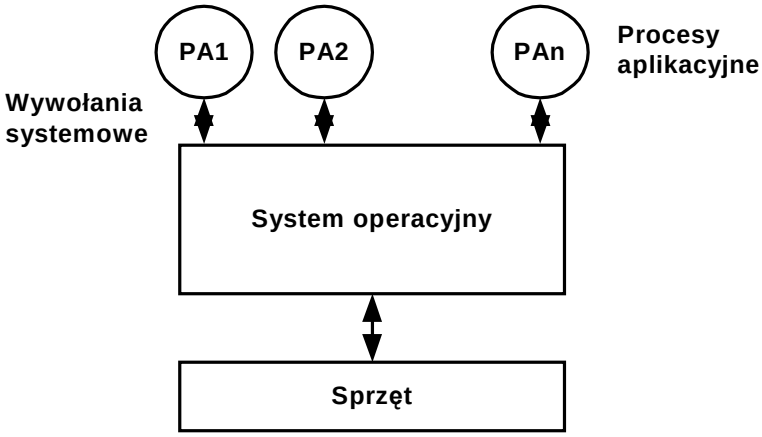
# 1 SYSTEM QNX 6 Neutrino

## 1.1 Wiadomości ogólne o systemach operacyjnych

Funkcje systemu operacyjnego:

- Zarządzanie zasobami systemu
  - zarządzanie procesami,
  - obsługa urządzeń,
  - obsługa pamięci wirtualnej
  - obsługa komunikacji
- Zapewnienie abstrakcyjnego dostępu do usług systemu
- Zapewnienie bezpieczeństwa

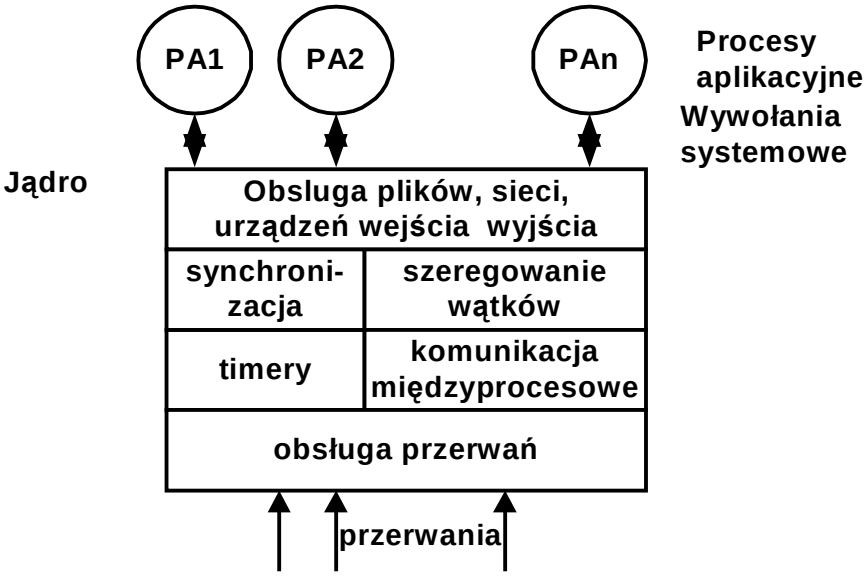



Rysunek 1-1 Rola systemu operacyjnego

Podstawowe struktury systemów operacyjnych:

- system monolityczny
- system z mikrojądrem.

W systemie monolitycznym podstawowe funkcje systemu umieszczone są w pojedynczym module programowym zwanym jądrem.




Rysunek 1-2 Jądro monolityczne

Fragmenty kodu jądra wykonywane są:
- pod wpływem przerwań
- uruchamiania wywołań systemowych.

Cechy systemu z jądrem monolitycznym:
- Jądro nie podlega szeregowaniu
- Awaria w jego obrębie skutkuje awarią całego systemu.
- Wysoka szybkość działania

Systemy monolityczne: Linux, RTLinux.

## 1.2 System QNX6 Neutrino

Wielozadaniowy, sieciowy system operacyjny czasu rzeczywistego

Podstawowe własności:

1. Architektura mikrojądra. System jest zbiorem procesów wymieniających komunikaty w środowisku lokalnym i rozproszonym.
2. System od podstaw zaprojektowany został jako wieloplatformowy: Intel x86, MIPS, PowerPC, SH-4, ARM, StrongARM i xScale.
3. System spełnia wymagania stawiane systemom czasu rzeczywistego, posiada wywłaszczającą strategię szeregowania i rozwinięty mechanizm priorytetów.
4. Zapewnia wysoką zgodność ze standardami POSIX 1003.1. Zawiera większość zawartych tam mechanizmów czasu rzeczywistego tym wątki.
5. Zapewnia wsparcie dla maszyn wieloprocesorowych SMP.
6. Zawiera wiele mechanizmów tolerowania awarii, w tym pakiet wspierania wysokiej dostępności HAM (*ang. High Availability Manager*).
7. Możliwe jest zastosowanie wielu systemów plików: RAM, Flash, QNX, Linux, DOS, CD-ROM, NFS.
8. Zapewnia wparcie dla wielu protokołów komunikacyjnych, w tym IPv4, IPv6.

POSIX - Standard na interfejs pomiędzy systemem operacyjnym a aplikacją. Określa:

- Niezbędny zestaw wywołań systemowych
- Postać funkcji systemowych (język C)

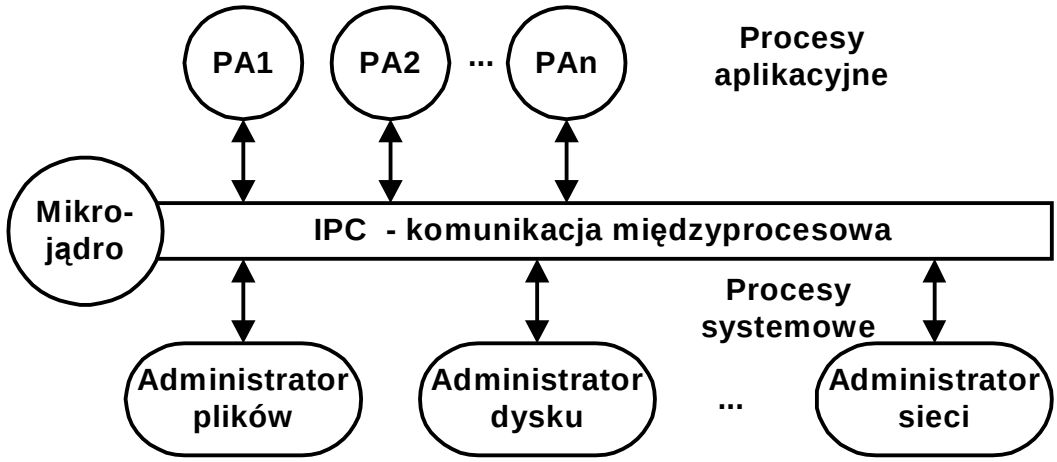
Wady systemu:

1. Wysoka cena
2. Utrata zgodności z istniejącym oprogramowaniem dla systemu QNX4
3. Wsparcie tylko dla ograniczonej liczby urządzeń peryferyjnych
4. Mała ilość oprogramowania użytkowego

## 1.3 Budowa systemu

Budowa systemu:

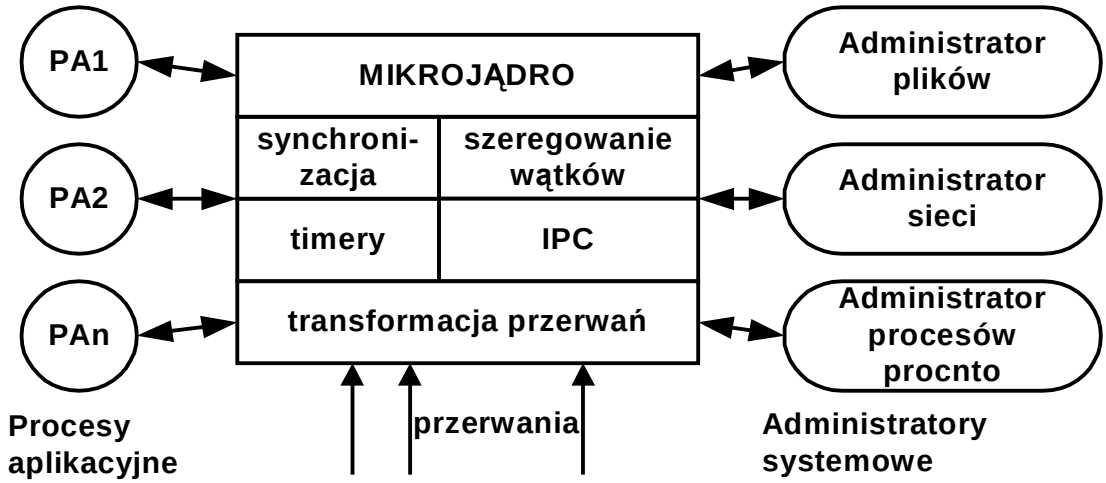
- Mikrojądro - microkernel
- Procesy systemowe - system processes
- Procesy aplikacyjne - application processes




Rysunek 1-3 Struktura systemu QNX6 Neutrino

Funkcje mikrojądra:

1. Implementacja podstawowych mechanizmów komunikacji międzyprocesowej: komunikatów, impulsów, zdarzeń, sygnałów.
2. Implementacja funkcji synchronizacji wątków takich jak muteksy, semafory, zmienne warunkowe, bariery, blokady, operacje atomowe.
3. Szeregowanie – procesy i wątki szeregowane są przez mikrojądro zgodnie z dostępnymi algorytmami szeregowania: FIFO, karuzelowy, sporadyczny.
4. Implementacja czasomierzy (*ang. Timers*).
5. Obsługa przerwań.




Rysunek 1-4 Struktura mikrojądra

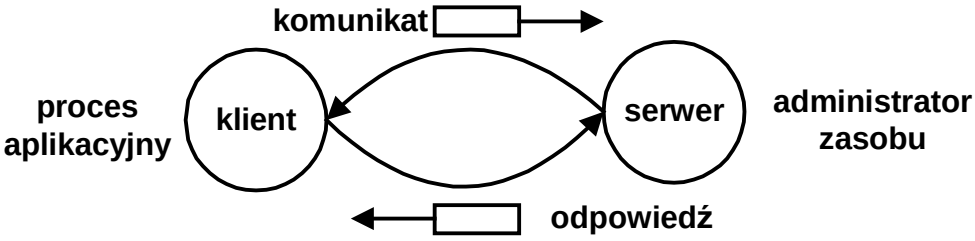
Zalety systemu z mikrojądrem:

1. Niezależne szeregowanie procesów systemowych - obsługa urządzeń wejścia / wyjścia - procesy, które podlegają zwykłemu szeregowaniu. Mechanizm umożliwia osiągnięcie lepszych charakterystyk czasowych systemu - ważne w systemach RTS.
2. Modularność - prowadzi do zwiększenia niezawodności.
3. Wzajemna izolacja procesów - każdy z procesów systemowych wykonywany jest w oddzielnie chronionym segmencie przestrzeni adresowej. Awaria jednego z procesów nie powoduje awarii innego procesu.
4. Możliwość dynamicznego uruchamiania procesów systemowych - procesy systemowe są zwykłymi procesami.

## 1.4 Komunikaty i komunikacja międzyprocesowa

Przesłanie komunikatu (*ang. message passing*)

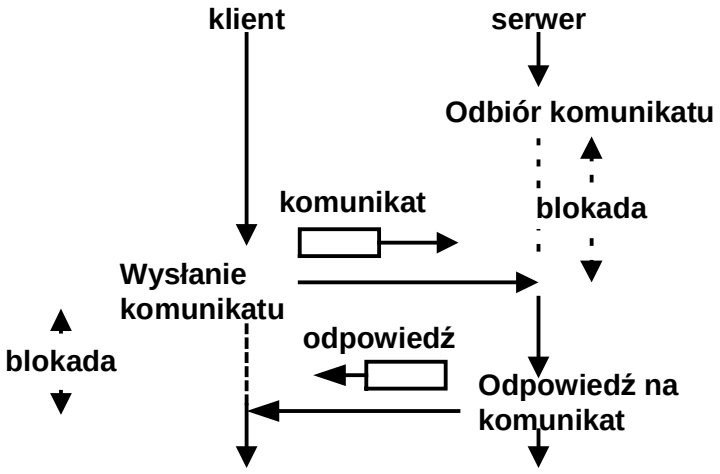
Przesłanie komunikatu pomiędzy procesami jest przesłaniem pomiędzy nimi pewnej liczby bajtów według ustalonego protokołu. Przesłanie komunikatu jest operacją atomową.




Rysunek 1-5 Klient przesyła komunikat do serwera

Przesłanie komunikatu składa się z trzech faz:

1. Wysłania komunikatu od procesu klienta do serwera. Proces klienta ulega zablokowaniu a komunikat odblokowuje proces serwera (o ile był zablokowany).
2. Serwer przetwarza komunikat i przesyła odpowiedź do klienta.
3. Proces klienta po otrzymaniu odpowiedzi ulega odblokowaniu.




Rysunek 1-6 Transakcja przesłania komunikatu

Mechanizmy komunikacji międzyprocesowej implementowane w mikrojądrze:

1. Komunikaty i impulsy - (*ang. messages, pulses*)
2. Sygnały - (*ang. signals*)
3. Zegary - (*ang. clocks*)
4. Czasomierze - (*ang. timers*)
5. Procedury obsługi przerwań - (*ang. interrupt handlers*)
6. Semafory - (*ang. semaphores*)
7. Blokady wzajemnego wykluczania - muteksy (*ang. mutual exclusion lock* )
8. Zmienne warunkowe (*ang. conditional variables*)
9. Bariery (*ang. barriers*)

Mechanizmy IPC zewnętrzne względem mikrojądra

1. Łącza nienazwane - (*ang. pipes*),
2. Łącza nazwane - (*ang. named pipes*)
3. Wspólna pamięć - (*ang. shared memory*)
4. Kolejki komunikatów - (*ang. message queues*)

## 1.5 Administratory zasobu i procesy systemowe

Administratory zasobu zapewniają jednolity dostęp do różnego rodzaju urządzeń.

Urządzenia rzeczywiste:
- dyski,
- porty szeregowe,
- porty równoległe
- sieć

Urządzenia wirtualne:
- sieciowy system plików
- okno
- pseudo terminal

Administrator zasobu jest procesem serwerowym który akceptuje komunikaty od innych procesów.

Administrator zasobu obsługuje zlecenia klientów odnoszące się do abstrakcji pliku.

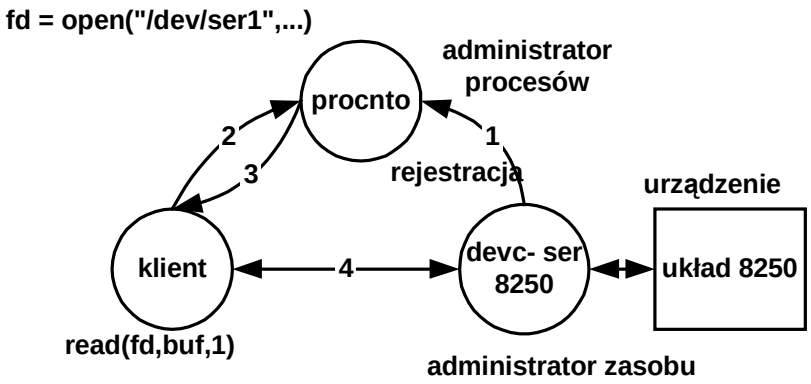
Plik jest abstrakcyjną strukturą na której można przeprowadzać operacje: otwarcie, odczyt, zapis, zamknięcie, zmianę bieżącej pozycji, ustawianie praw dostępu.

Każde urządzenie w systemie QNX6 Neutrino widziane jest jako plik specjalny.

Rodzaje plików:
- pliki specjalne (urządzenia, gniazdka, kolejki, semafory, segmenty pamięci dzielonej)
- pliki regularne
- katalogi

| | | |
|---|---|---|
| **/dev/hd0** | **devb-eide** | Proces obsługi dysków stałych typu IDE |
| **/dev/fd0** | **devb-fdc** | Proces obsługi stacji dyskietek |
| **/dev/par1** | **devc-par** | Proces obsługi portu równoległego |
| **/dev/ser1** | **devc-ser8250** | Proces obsługi portu szeregowego RS232 |
| **/dev/con** | **devc-con** | Proces obsługi konsoli |
| **/** | **fs-qnx4.so** | Proces obsługi systemu plików QNX4 |

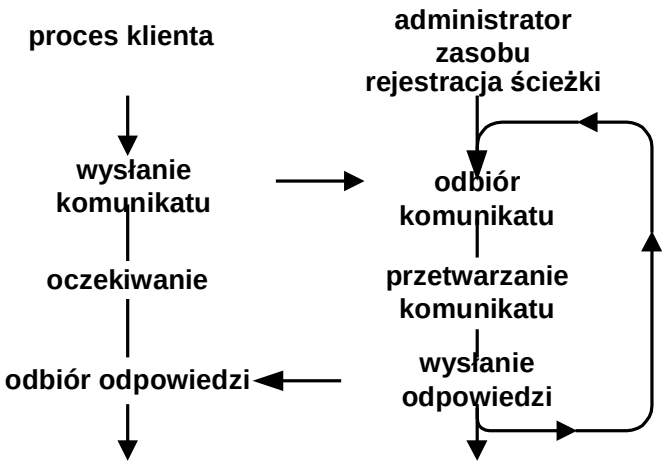
Tabela 1-1 Ścieżki do urządzeń i obsługujące je administratory




**Rysunek 1-7 Współpraca procesu klienta z portem transmisji szeregowej.**

Współpraca procesu klienta z portem transmisji szeregowej:

1. Przy starcie systemu proces devc-ser8250 rejestruje się w administratorze procesów procnto.
2. Proces klienta wysyła komunikat do lokalnego administratora procesów procnto z zapytaniem w gestii jakiego procesu leży ścieżka /dev/ser1.
3. Lokalny administrator procesów odpowie że należy się kontaktować z procesem devc-ser8250.
4. Proces klienta wysyła komunikaty do procesu devc-ser8250.




Rysunek 1-8 Współpraca pomiędzy procesem klienta  a administratorem zasobu

| | |
|---|---|
| **procnto** | Zarządzanie pamięcią, wątkami i procesami |
| **devb-eide** | Proces obsługi dysków stałych typu IDE |
| **devb-fdc** | Proces  obsługi stacji dyskietek |
| **devc-par** | Proces  obsługi portu równoległego |
| **devc-ser8250** | Proces  obsługi portu szeregowego RS232 |
| **devc-con** | Proces  obsługi konsoli |
| **io-net** | Ogólny administrator sieci |
| **io-graphisc** | Proces  obsługi karty graficznej |
| **Photon** | Proces  interfejsu graficznego |

Tabela 1-2 Ważniejsze procesy systemowe systemu QNX6 Neutrino

## 1.6 System plików

W systemie QNX6 Neutrino zaimplementowano wiele różnych systemów plików. Każdy z systemów plików obejmuje fragment przestrzeni nazw i obsługuje drzewo katalogów i plików leżące poniżej punktu jego montowania.

Własności:

1. Systemy plików mogą być startowane i zatrzymywane dynamicznie w trakcie pracy systemu.
2. Równocześnie może współistnieć wiele systemów plików.
3. System plików wykonywany na jednym węźle może być w przeźroczysty sposób dostępny z innego węzła.

Najważniejsze systemy plików dostępne w QNX6 Neutrino:.

1. Bezpośredni system plików - Stosowany jest w wielu sterownikach wbudowanych.

2. System plików RAM - prosty system plików pozwalający na zapis i odczyt plików umieszczonych w pamięci RAM w katalogu `/dev/shmem`.

3. System plików QNX4 - jest to podstawowy system plików. Charakteryzuje się wysoką odpornością na awarie, wydajnością i architekturą wielowątkową, obsługiwany przez proces **`fs-qnx4.so`**.

4. System plików DOS - System plików jest obsługiwany przez proces **`fs-dos.so`**.

5. System plików CD-ROM - Obsługuje format ISO9660 System obsługiwany przez proces **`fs-cd.so`**.

6. System plików FFS3 - system plików przeznaczony jest dla pamięci Flash typu NOR (Compact Flash i Smart Media). System zawiera zabezpieczenia chroniące przed utratą integralności w przypadku wyłączenia zasilania w trakcie operacji zapisu. Stosowany jest w sterownikach wbudowanych.

7. Sieciowy system plików NFS - Stacja kliencka może sięgać poprzez sieć do plików położonych na zdalnym serwerze. System oparty jest na technologii RPC (*ang. Remote Procedures Calls*). Jako warstwę transportowa stosuje protokół TCP/IP. Procesem obsługującym NFS jest **`fs-nfs2`** lub **`fs-nfs3`**.

8. Sieciowy system plików CIFS - CISF jest sieciowym systemem plików wspieranym przez Microsoft (uprzednio SMB). Stacja kliencka może uzyskać dostęp poprzez sieć do systemów plików na serwerze Windows lub Linux na którym uruchomiono serwer SMB. Jako warstwę transportowa stosowany jest protokół TCP/IP a system utrzymywany jest przez proces **`fs-cifs`**.

9. System plików Ext2 - system plików Ext2 jest używany przez Linux'a. Procesem obsługującym ten system plików jest fs-ext2.so.

## 1.7 Onet – rodzima sieć komunikacyjna systemu QNX6 Neutrino

Unikalność mikrojądra Neutrino polega na łatwości implementacji środowisk rozproszonych (*ang. distributed enviroments*). Osiągnięto to dzięki paradygmatowi przekazywania komunikatów, który dla pracy lokalnej i sieciowej pozostaje zasadniczo taki sam.

Fundamentem tego systemu – protokół Qnet umożliwia jednolitą metodę dostępu do zasobów, tak lokalnych jak i zdalnych.

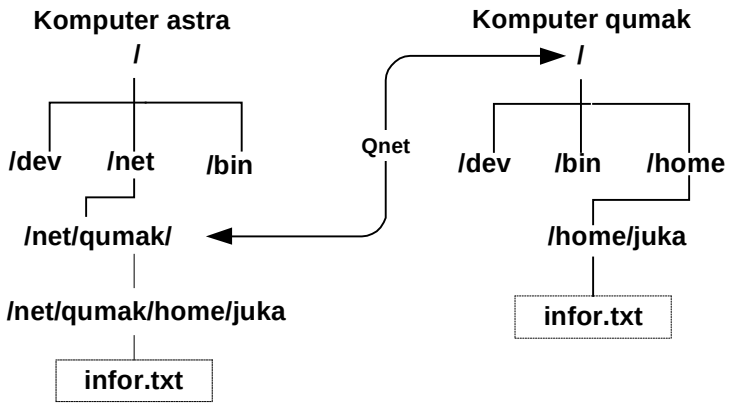
Standardowe programy narzędziowe operujące na takich zasobach systemu jak pliki działają tak samo dla zasobów lokalnych i zdalnych.

Zastosowanie sieci Qnet pozwala na osiągnięcie następujących korzyści:

1. Przeźroczysty dostęp do zdalnego systemu plików.
2. Łatwość budowy aplikacji skalowalnych.
3. Łatwość tworzenia aplikacji rozproszonych składających się z komunikujących się procesów.
4. Łatwość tworzenie aplikacji równoległych wykonywanych na połączonych siecią komunikacyjną komputerach.

Przykład:

```
$ cat /net/qumak/home/juka/infor.txt
```

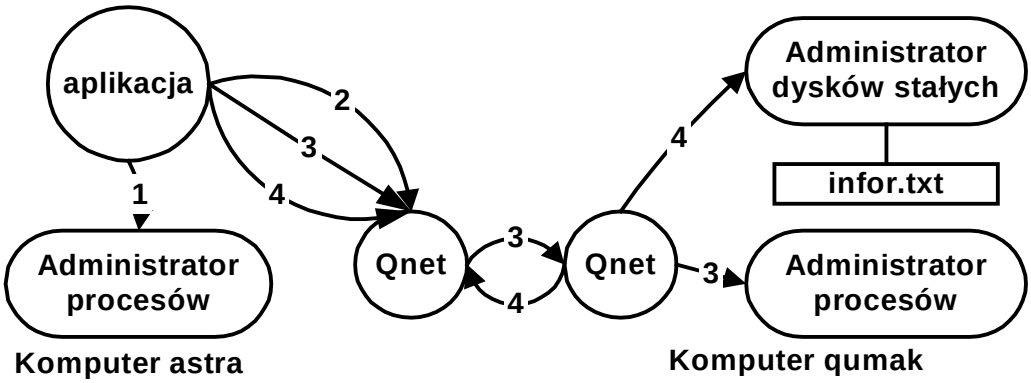



**Rysunek 1-9 Odwzorowanie przestrzeni nazw komputera qumak w katalogu /net/qumak komputera astra**

```
fh = open("/net/qumak/home/juka/infor.txt", O_RDWR)
```

| Z komputera astra | Lokalnie | Zasób |
|---|---|---|
| **/net/qumak/dev/ser1** | **/dev/ser1** | Port szeregowy |
| **/net/qumak/dev/par** | **/dev/par** | Port równoległy |
| **/net/qumak/home/** | **/home** | Katalog |
| **/net/qumak /home/juka/infor.txt** | **/home/juka/infor.txt** | Plik regularny |

Tabela 1-3 Zasoby komputera **qumak** lokalnie i poprzez sieć




Rysunek 1-10 Dostęp klienta do zasobu położonego na komputerze zdalnym

Katalog sieciowy - odwzorowuje nazwy komputerów połączonych siecią Standardowo katalog sieciowy nazywa się **/net**.

Zawartość katalogu sieciowego a tym samym nazwy komputerów pracujących w sieci Qnet uzyskać można poleceniem:

```
$ls /net
$ astra qumak
```

Sieć Qnet obsługuje wielokrotne połączenia sieciowe. Z połączeniami wielokrotnymi łączy się pojęcie QoS (*ang. Quality of Service*)

- połączenie równoważące obciążenie,
- połączenie preferowane,
- połączenie wyłączne.