Systemy czasu rzeczywistego

Mgr Radosław Gołąb
Zakład Sieciowych Systemów Informatycznych
Państwowa Wyższa Szkoła Zawodowa w Krośnie
ul. Wyspiańskiego 20 38-400 Krosno
e-mail: golab@data.pl

STRESZCZENIE

Przedmiotem referatu jest przedstawienie zagadnień dotyczących systemów operacyjnych spełniających wymagania czasu rzeczywistego. W referacie poruszono zagadnienia związane z zastosowaniem, specyfiką, oraz architekturą (na przykładzie systemu QNX) omawianych systemów.

SUMMARY

The subject of the present paper is to discuss the problems concerning the operating systems fulfilling the requirements of the real time. In the thesis we deal with the application, the nature and the architecture of the systems in question (taking for the example the QNX system).

1. Wstęp

System czasu rzeczywistego (*real-time system*), to system komputerowy, w którym obsługiwanie zdarzeń dokonuje się w z góry przewidzianych limitach czasu. Rozróżnia się systemy czasu rzeczywistego pobudzane zdarzeniami oraz systemy pobudzane czasem. W systemie czasu rzeczywistego występują czujniki odbierające bodźce z otoczenia oraz aktywatory. Czujniki mogą wytwarzać impulsy okresowo, nieokresowo lub sporadycznie. Wymagania odnośnie czasu reakcji systemu mogą być różnie formułowane, w zależności od wymagań stawianych systemowi. Sposób ich spełniania pozwala na określenie podstawowej klasyfikacji tych systemów. Systemy te dzielimy na systemy o twardych (hard real-time) oraz miękkich (soft real-time) wymaganiach czasowych. W miękkim systemie czasu rzeczywistego dopuszcza się okazjonalnie przekraczanie limitów czasu reakcji. Natomiast rygorystyczny system czasu rzeczywistego o twardych wymaganiach czasowych to taki system, w którym nie może nastąpić ani jedno przekroczenie limitu czasu reakcji na zdarzenie. [1]

Główne dziedziny zastosowań systemów czasu rzeczywistego to:

- systemy sterowania i monitorowania (SCADA-Supervisory Control and Data Acquisition) np. sterowanie procesem produkcji w fabryce;

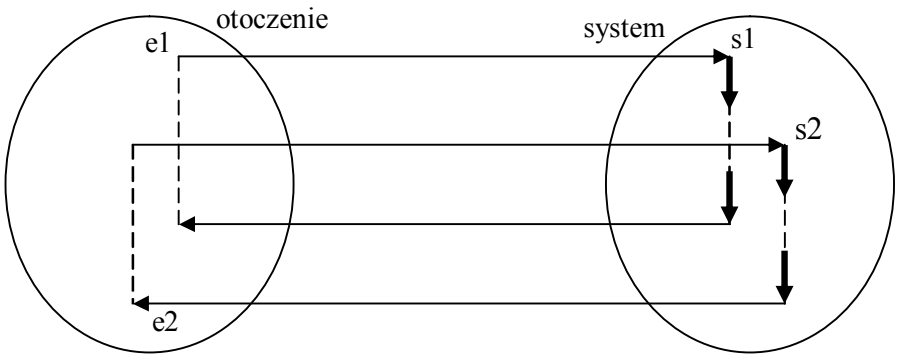
- systemy bieżącego przetwarzania transakcji (OLTP- On-Line Transaction Processing) np. obsługa transakcji dokonywanych za pomocą kart kredytowych;

- systemy bezwzględnego nadzoru (MCS- Mission Critical Systems) np. system kontroli funkcjonowania elektrowni jądrowej.

Specyficzną cechą omawianych systemów jest żądanie nieprzerwanego działania od momentu zainicjowania do fazy usunięcia. Wymagania te wpływają w istotny sposób na fazy konstrukcji, rozbudowy i eksploatacji oprogramowania, wymuszają rozbudowaną obsługę sytuacji wyjątkowych, czyli reakcji podejmowanych przez system w przypadku wykrycia stanów awaryjnych. Wymienione cechy definiują odrębną specyfikę systemów czasu rzeczywistego, ściśle określając ich obszar zastosowań we współczesnej informatyce.

2. Systemy czasy rzeczywistego (podstawowe pojęcia)

System czasu rzeczywistego jest analizowany i opisywany w kontekście dwóch współdziałających procesów: otoczenie i system komputerowy, przy czym działanie tego ostatniego jest dodatkowo uzależnione od czasu. [3]



Rys 1 Powiązanie systemu czasu rzeczywistego z otoczeniem

Rysunek przedstawia system, pracujący w fizycznym **otoczeniu**, które **zmienia się autonomicznie w miarę upływu czasu**. Wynika stąd podstawowa cecha systemów czasu rzeczywistego, tzn. **reagowalność**. Jest to wymaganie właściwego reagowania na zdarzenia zachodzące w otoczeniu, nawet w przypadkach wystąpienia w systemie błędów. Jako przykład można przytoczyć awarię programu edytora tekstu. W wyniku awarii może nastąpić utrata pliku lub jego kopii, skutki te chociaż dokuczliwe dla użytkownika, nie są jednak niebezpieczne dla stabilności systemu. Jeżeli błąd nastąpi jednak np. w programie automatycznego pilota, i wystąpi niewłaściwe sterowanie, to skutki mogą być tragiczne, ponieważ działanie pola grawitacyjnego nie może być kontrolowane.

Wynika stąd, iż nawet w wypadku wykrycia błędów, ich usunięcie i prawidłowa reakcja na ewolucję świata zewnętrznego (otoczenia) musi być zrealizowana w ściśle określonym czasie mniejszym niż minimum określone w wymaganiach konstrukcyjnych systemu. Nie można bowiem zatrzymać procesów zachodzących w otoczeniu na czas wykrycia błędu i poprawienia oprogramowania. Nasuwa się zatem wniosek, iż w czasie projektowania systemów czasu rzeczywistego nie można pozwolić sobie na ich tworzenie i rozwijanie metodą „domową" tzn. rozpoczęcie tworzenia od pierwszej wersji i następnie przeprowadzanie analizy i poprawiania, gdyż takie postępowanie z uwagi na brak możliwości usystematyzowania może zakończyć się niepowodzeniem.

Dalsze rozważania zostaną oparte na definicji systemu czasu rzeczywistego (zgodnie z IEEE/ANSI) która brzmi:

System czasu rzeczywistego *jest to system komputerowy, w którym obliczenia są wykonywane współbieżnie z procesem zewnętrznym (otoczenie) w celu sterowania, nadzorowania lub terminowego reagowania na zdarzenia występujące w tym procesie (otoczeniu).*

Przytoczoną definicję można traktować jako podsumowanie wcześniejszych rozważań. Opierając się na niej można wyodrębnić dwa niezależne procesy:
a.) proces określony przez obliczenia realizowane w systemie komputerowym
b.) proces zachodzący w otoczeniu, które dostarcza informacji wejściowych systemowi komputerowemu (zdarzenia na które reaguje system) i jest obszarem, gdzie kierowane są wyniki obliczeń (reakcje na zdarzenia).
W trakcie tworzenia systemów czasu rzeczywistego należy barć pod uwagę różne wymagania określające pożądane cechy systemu. Wyróżnić można pięć cech charakterystycznych:

1. **Ciągłość działania**. System powinien pracować bez przerw. W czasie eksploatacji systemu od jego wdrożenia do wycofania system powinien działać bez przerwy. Można więc powiedzieć że obliczenia systemu są nieskończone i trudno jest w nim wyodrębnić stan początkowy. Najistotniejsze są stany oczekiwania na występujące w otoczeniu zdarzenia.

2. **Zależność systemu od otoczenia**. System czasu rzeczywistego rozpatrywany jest kontekście otoczenia, co oznacza że działanie systemu (obliczenia) zależne są od zdarzeń i danych generowanych przez procesy zewnętrzne w stosunku do systemu (otoczenie).

3. **Współbieżność**. Środowisko w którym działa system czasu rzeczywistego (otoczenie) składa się z wielu obiektów (podsystemów), które działają względem siebie współbieżnie, generując dane lub zdarzenia wymagające obsługi przez system. Wymagania jednoczesnej obsługi narzucają współbieżność systemu, tak więc system składa się z szeregu procesów współbieżnych, które dostarczają mechanizmów obsługi zdarzeń.

4. **Przewidywalność**. Zdarzenia i dane obsługiwane przez system czasu rzeczywistego pojawiają się w momentach przypadkowych, jednoczesne wystąpienia zdarzeń może wymagać ich jednoczesnej obsługi, co przy ograniczonych zasobach prowadzi do współzawodnictwa o dostęp do nich. Stąd system mimo że złożony z szeregu procesów współbieżnych (co jest związane ze strukturą wewnętrzną systemu), na zewnątrz musi reagować wg. założonych wymagań.

5. **Punktualność**. Odpowiedzi systemu (reakcje na zdarzenia zachodzące w otoczeniu) winny być obliczane zgodnie z zaprojektowanymi algorytmami i dostarczane do otoczenia w odpowiednich momentach czasowych. Natura otoczenia, tzn. brak możliwości zatrzymania procesu zewnętrznego stawia dodatkowe wymagania, by oprócz poprawności uzyskanych wyników, moment ich przekazania do otoczenia spełniał określone wymagania. Wyniki dostarczone w nieodpowiednim czasie mogą doprowadzić do niepoprawnego działania czy nawet upadku systemu. [4]

3. Charakterystyka systemu czasu rzeczywistego QNX.

Pierwowzorem systemu QNX był system operacyjny THOTH będący wynikiem prac badawczych dotyczących technik komunikacji międzyprocesorowej prowadzonych na Uniwersytecie Waterloo. Dwóch pracowników uniwersytetu zaprojektowało i zaimplementowało dla mikrokomputera IBM PC nowy system operacyjny, który został nazwany pierwotnie QUNIX, następnie QNX i pod tą nazwą funkcjonuje do dzisiaj.
W roku 1982 roku QNX v.1.0 by pierwszym wielozadaniowym i wielodostępnym systemem operacyjnym dla mikrokomputera IBM PC. Cechy takie jak: wielozadaniowość, wielodostępność, dedykowalność dla architektury IBM AT i zintegrowana sieć lokalna stały się przyczyną zdobycia przez system QNX dużej popularności na całym świecie.

Obecnie QNX jest całkowicie zgodny ze standardem POSIX (**P**ortable **O**perating **S**ystem **I**nterface for Computer Environment). Zdefiniowany przez IEEE (ang. Institute of Electrical and Electronic Engineers) POSIX określa interfejs pomiędzy systemem operacyjnym a światem zewnętrznym (aplikacje i użytkownicy). Wiele przemawia za tym, że POSIX może stać się uniwersalną platformą dla Systemów Otwartych (ang. Open Systems). Np. coraz częściej różne instytucje wybierając system operacyjny do własnych zastosowań, stawiają warunek zgodności ze standardem POSIX.

Zgodność ze standardem POSIX otwiera przed systemem QNX nowe możliwości:

- duża ilość dostępnych aplikacji (zgodność na poziomie kodu źródłowego pozwala na bezproblemową implemetację);

- zwiększa się ilość użytkowników mogących bez przygotowania wykorzystywać system (zgodność ze środowiskiem użytkownika systemu UNIX: tryb graficzny, shell, polecenia, biblioteki C, itp.);

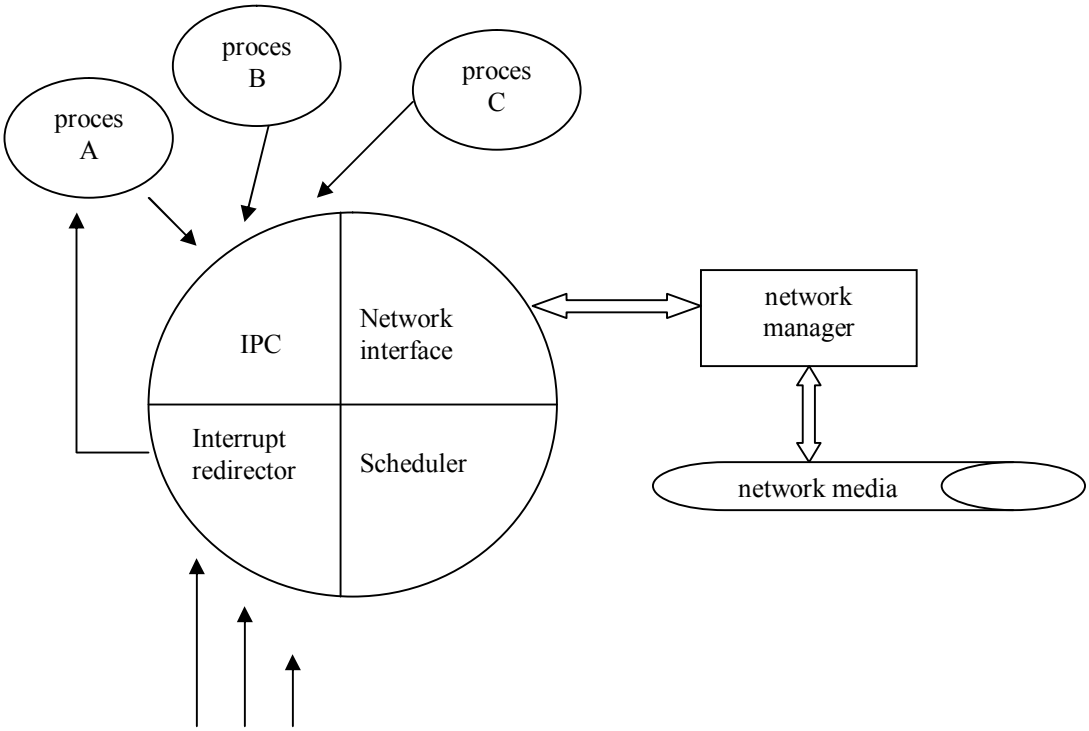
- zwiększa się możliwość wykorzystania systemu w różnych zastosowaniach i przez różnych niekoniecznie przeszkolonych użytkowników. [5]

3.1. Budowa systemu QNX

Architektura systemu QNX oparta jest o model client-server, w przeciwieństwie do systemu UNIX, którego struktura oparta jest o monolityczne jądro. System QNX składa się z wielu modułów zwanych zadaniami administrującymi. Moduły te współpracują ze sobą za pośrednictwem mikrojądra wykorzystując do tego efektywną technik komunikacji zwaną przesyłaniem komunikatów (message passing). Modułowa struktura pozwala na dynamiczne konfigurowanie systemu w zależności od potrzeb zewnętrznych. Zintegrowanie usług sieci lokalnej z systemem operacyjnym pozwala rozszerzyć komunikację międzyzadaniową na całą sieć.

Do zadań mikrojądra należy:

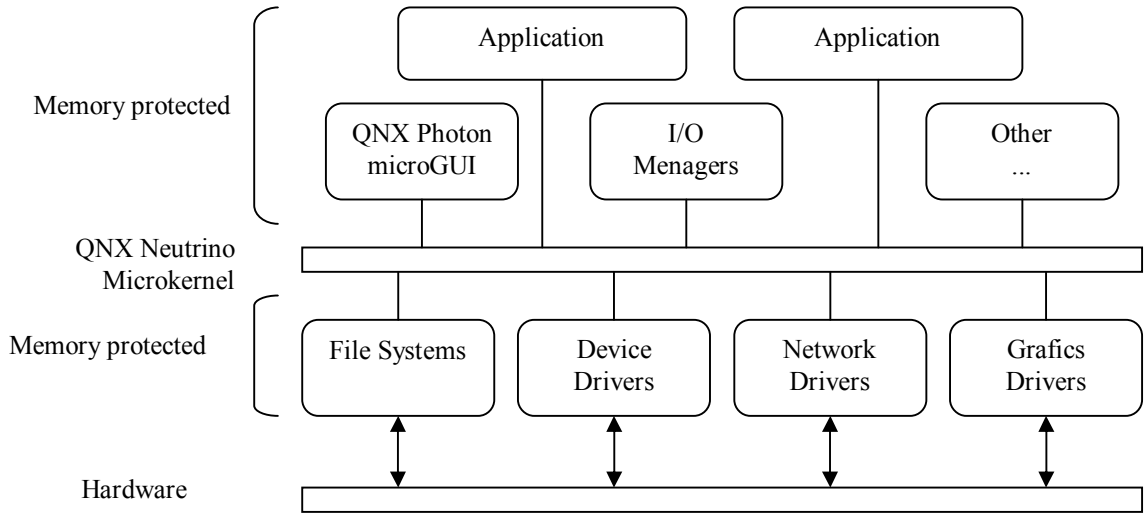
- Przekazywanie komunikatów (IPC);
- Szeregowanie procesów;
- Odbieranie przerwań;
- Niskopoziomowa komunikacja sieciowa;
- Przekazywanie procesom sygnałów;



Rys. 2 Budowa systemu QNX

Najważniejszą częścią systemu QNX jest mikrojądro które dostarcza podstawowych mechanizmów do obsługi w czasie rzeczywistym aplikacji wbudowanych, takich jak przekazywanie komunikatów, obsługa wątków POSIX, mutex-ów, zmiennych warunkowych, semaforów, sygnałów i szeregowania procesów. Jak powiedziano wcześniej system QNX charakteryzuje się modularną budową co umożliwia mechanizm wymiany komunikatów tworzący programową magistralę, pozwalającą na przyłączenie lub odłączenie jakichkolwiek potrzebnych modułów systemowych bez konieczności restartowania systemu. System ten charakteryzuje się ogromną skalowalnością od niewielkich urządzeń, poprzez wieloprocesorowe maszyny do ogromnych systemów rozproszonych.

System operacyjny czasu rzeczywistego korzysta z prostej architektury pamięci, w której trudne do wykrycia błędy programistyczne, typu wadliwy wskaźnik w języku C, może spowodować, że programy nadpiszą się nawzajem lub uszkodzą jądro co może spowodować błąd systemu. System bazujący na QNX-ie może inteligentnie opanować błędy oprogramowania, nawet w sterownikach urządzeń i innych krytycznych programach bez konieczności przeładowania, ponieważ każdy komponent systemu uruchomiony jest w swojej własnej chronionej przestrzeni adresowej MMU. [5]



Rysunek 3 Pamięć chroniona w systemie QNX

Dziedzina zastosowań systemów czasu rzeczywistego jest ogromna, od prostych systemów wbudowanych stanowiących elementy większej całości po systemy o wiele bardziej złożone np. systemy nawigacji, nadzoru produkcji, dowodzenia itp. Obszar ich zastosowań ciągle ulega powiększeniu, o czym mogą świadczyć np. zastosowania systemów czasu rzeczywistego w branży telekomunikacyjnej, m.in. telefonia komórkowa. Powoduje to powstawanie coraz to nowych zastosowań dla tych systemów stanowiących ciekawe wyzwanie dla naukowców.

LITERATURA

[1] Sacha K., Systemy czasu rzeczywistego, Wyd. 2 (zmienione), Oficyna Wydawnicza PW, Warszawa, 1999.
[2] Sacha K., QNX – System operacyjny, X-Serwis, Warszawa, 1995.
[3] Szmuc T., Motet G., Specyfikacja i projektowanie oprogramowania systemów czasu rzeczywistego, Wydawnictwo AGH, Kraków, 2000.

[4] Szmuc T., Modele i metody inżynierii oprogramowania systemów czasu rzeczywistego, Wydawnictwo AGH, Kraków, 2001
[5] www.qnx.com