Systemy Czasu Rzeczywistego (SCR)

Wykład 12: Elementy systemów operacyjnych czasu rzeczywistego (1/2)

WYDZIAŁ ELEKTROTECHNIKI I AUTOMATYKI – KATEDRA INŻYNIERII SYSTEMÓW STEROWANIA
Kierunek: Automatyka i Robotyka
Studia stacjonarne I stopnia: rok II, semestr IV
dr inż. Tomasz Rutkowski
2017

Podstawowy podział aplikacji czasu rzeczywistego oraz sposoby ich realizacji

© SCR 2017

Podział aplikacji czasu rzeczywistego

- **„Czysto cykliczne”**

  Każde zadanie wydzielone w ramach aplikacji wykonuje się cyklicznie oraz wymaga „prawie” tyle samo czasu. Dane wejściowe pobierane są periodycznie, dane wyjściowe generowane są również periodycznie.

  (np. system sterowania sygnalizacją świetlną)

- **„Prawie cykliczne”**

  Wszystkie zadania są wykonywane cyklicznie, z wyjątkiem odpowiedzi na zewnętrzne, zdarzenia przypadkowe.

  (np. system sterowania mikrofalówką)

© SCR 2017

Podział aplikacji czasu rzeczywistego

- **„Asynchroniczne i częściowo przewidywalne”**

  Większość zadań ma charakter asynchroniczny lub wymagany czas obliczeń znacznie się różni w każdym cyklu pracy systemu.

  Znane są ograniczenia na czas obliczeń w różnych sytuacjach oraz statystyki poszczególnych zdarzeń.

  (np. kodowanie i dekodowanie sygnału audio i video)

- **„Asynchroniczne i nieprzewidywalne”**

  Aplikacje tego typu reagują na asynchroniczne zdarzenia, w ich wyniku wykonują skomplikowane obliczenia, których czas może znacznie się różnić w różnych przypadkach.

  (np. system sterowania pojazdem w czasie rzeczywistym)

© SCR 2017

Sposoby realizacji aplikacji czasu rzeczywistego

- Sposoby realizacji zależą, między innymi od:
  - złożoności zadania, wielkości i trudności „problemu do rozwiązania”
  - kosztów
- A wiążą się, między innymi z:
  - typem urządzenia/urządzeń sterowania cyfrowego
  - systemy z super-pętlą (PLC) lub system operacyjny oparty o jądro wielozadaniowe (SOCR)
  - infrastrukturą sprzętową: systemy jednoprocesorowe, wieloprocesorowe, rozproszone
  - sposobem programowania systemu: równoległe, współbieżne, sekwencyjne
  - językiem programowania

© SCR 2017

Podstawy Systemów Operacyjnych

System Operacyjny Czasu Rzeczywistego

© SCR 2017

Czym jest System Operacyjny (SO)?

- System operacyjny jest zbiorem modułów oprogramowania, który nadzoruje wykonywanie programów i zarządza dostępnymi zasobami
- System operacyjny jest pośrednikiem między użytkownikiem komputera a sprzętem komputerowym
- System operacyjny jest programem sterującym różnymi urządzeniami wejścia-wyjścia oraz programami użytkownika

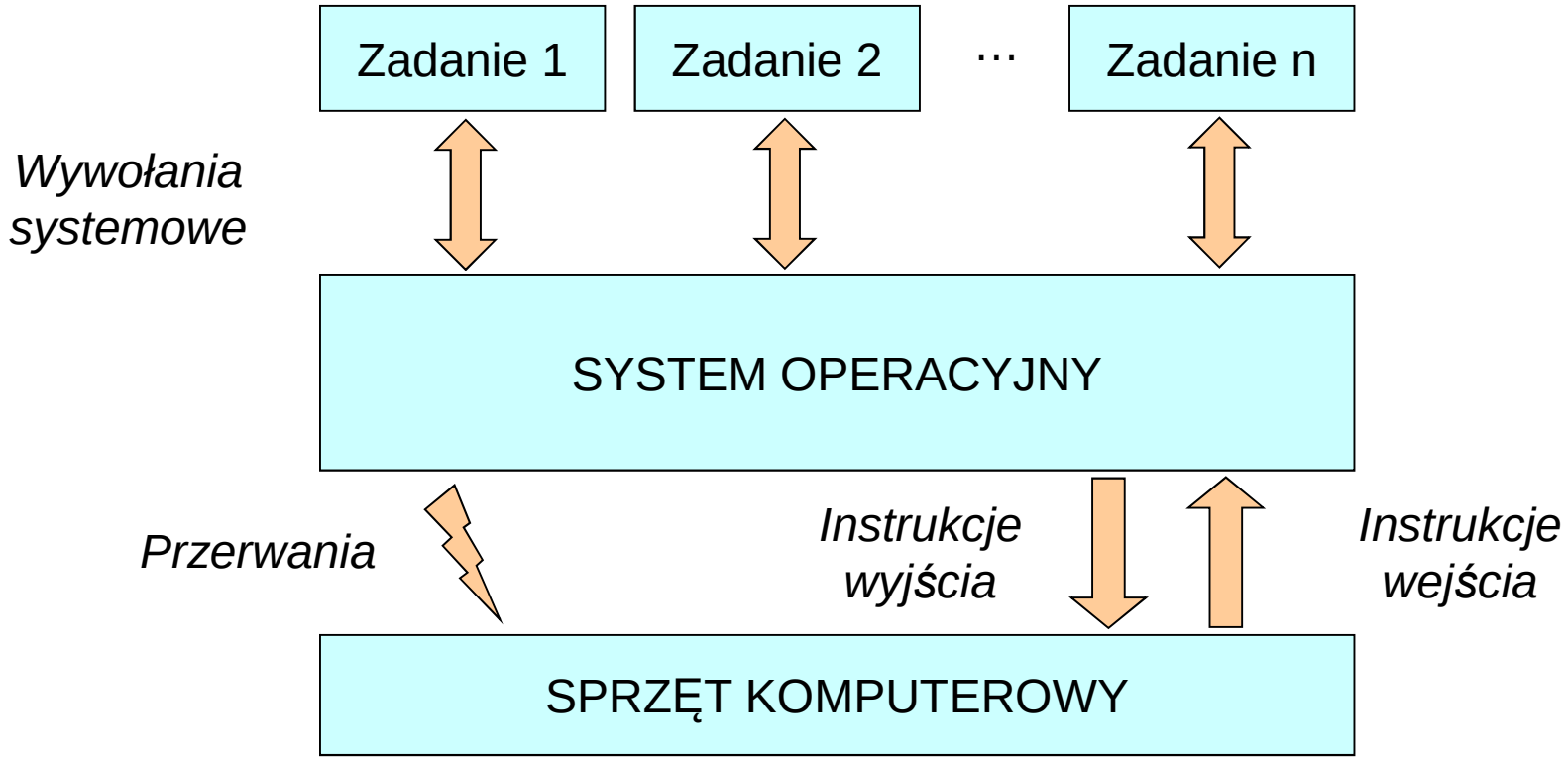
© SCR 2017

Czym jest System Operacyjny (SO)?

- System operacyjny, nazywany również jądrem, jest programem, który działa na komputerze nieustannie, pozostałe działające programy są programami (zadaniami) użytkownika
- System operacyjny nie wykonuje żadnej użytecznej pracy, tworzy natomiast środowisko w którym inne programy mogą ją wykonywać
- System operacyjny:
  - działa jako dystrybutor zasobów
  - przydziela zasoby poszczególnym programom i użytkownikom,
  - pełni rolę arbitra, gdy dochodzi do konfliktowych zamówień na zasoby
  - decyduje o przydziale zasobów mając na względzie wydajne i harmonijne działanie całego komputera

© SCR 2017

Czym jest System Operacyjny (SO)?



© SCR 2017

Najważniejsze funkcje Systemu Operacyjnego

- Implementacja współbieżności procesów i wątków
- Zarządzanie urządzeniami wejścia-wyjścia
- Implementacja pamięci wirtualnej
- Implementacja systemu plików
- Implementacja protokołów komunikacyjnych
- Implementacja interfejsu użytkownika
- Zapewnienie bezpieczeństwa

© SCR 2017

Typy Systemów Operacyjnych

- System interakcyjny
- System wsadowy
- System z podziałem czasu
- System czasu rzeczywistego
- System rozproszony
- System jednostanowiskowy

© SCR 2017

Typy Systemów Operacyjnych

- System jednozadaniowy *(np. DOS)*
- System wielozadaniowy *(np. Win XP, UNIX, Linux)*

- System jednodostępowy *(np. DOS, Win Me)*
- System wielodostępowy *(np. UNIX, Linux, Win XP)*

© SCR 2017

Ogólna klasyfikacja Systemów Operacyjnych

Systemy Operacyjne sterowane zdarzeniami

- ryzyko „przeciążenia” systemu w wyniku napływania dużej liczby danych i konieczności ich przetworzenia

Systemy Operacyjne sterowane czasem

- działają „w rytm” zegara czasu rzeczywistego,
- nie są podatne na „przeciążenia” wynikające z potrzeby przetworzenia dużej liczby danych

Protokół zarządzania obciążeniem – przydzielanie zadań i zasobów do wykonujących je jednostek tak aby w pełni wykorzystać możliwości systemu

© SCR 2017

Elementy Systemów Operacyjnych Czasu Rzeczywistego

Wymagania funkcjonalne stawiane systemom operacyjnym czasu rzeczywistego SOCR *(ang. Real Time Operating Systems, RTOS):*

- Determinizm
- Wysoka reaktywność
- Wysoki poziom kontroli użytkownika nad pracą systemu
- Wysoka wiarygodność świadczenia usług, zdolność do tolerowania awarii
- Małe zapotrzebowanie na zasoby
- Prosta konstrukcja i dobra dokumentacja
- Wsparcie dla różnych platform sprzętowych
- Dobre wsparcie od dostawcy i pozycja na rynku

© SCR 2017

Elementy Systemów Operacyjnych Czasu Rzeczywistego

Wymagane mechanizmy systemów operacyjnych czasu rzeczywistego SOCR:

- System musi umożliwić wykonanie wielu procesów wielowątkowych
- Wątki muszą charakteryzować się priorytetami
- Musi być stosowana wywłaszczająca strategia szeregowania
- System musi realizować mechanizm przewidywalnej synchronizacji wątków
- Musi istnieć dziedziczenie lub pułapy priorytetów
- System musi być deterministyczny
- System musi być pozbawiony błędów

© SCR 2017

Podstawowe różnice pomiędzy SOCR a „zwykłym” SO

Powszechnie wykorzystywane „zwykłe” SO:

- stosują liberalna politykę podziału zasobów
- priorytety różnych zadań mogą być w tych systemach dynamicznie zmieniane w sposób nieznany użytkownikowi
- procesy czasu rzeczywistego nie są specjalnie uprzywilejowane, nie są też zachowywane ich priorytety
- nie można przewidzieć, który z procesów zostanie wykonany w jakim czasie, ani w jakiej kolejności będą one przetwarzane
- faworyzują użytkowników terminalowych i programy interaktywne (np. edytor tekstu)
- procesy wykonywane w trybie systemowym nie mogą zostać wywłaszczone

© SCR 2017

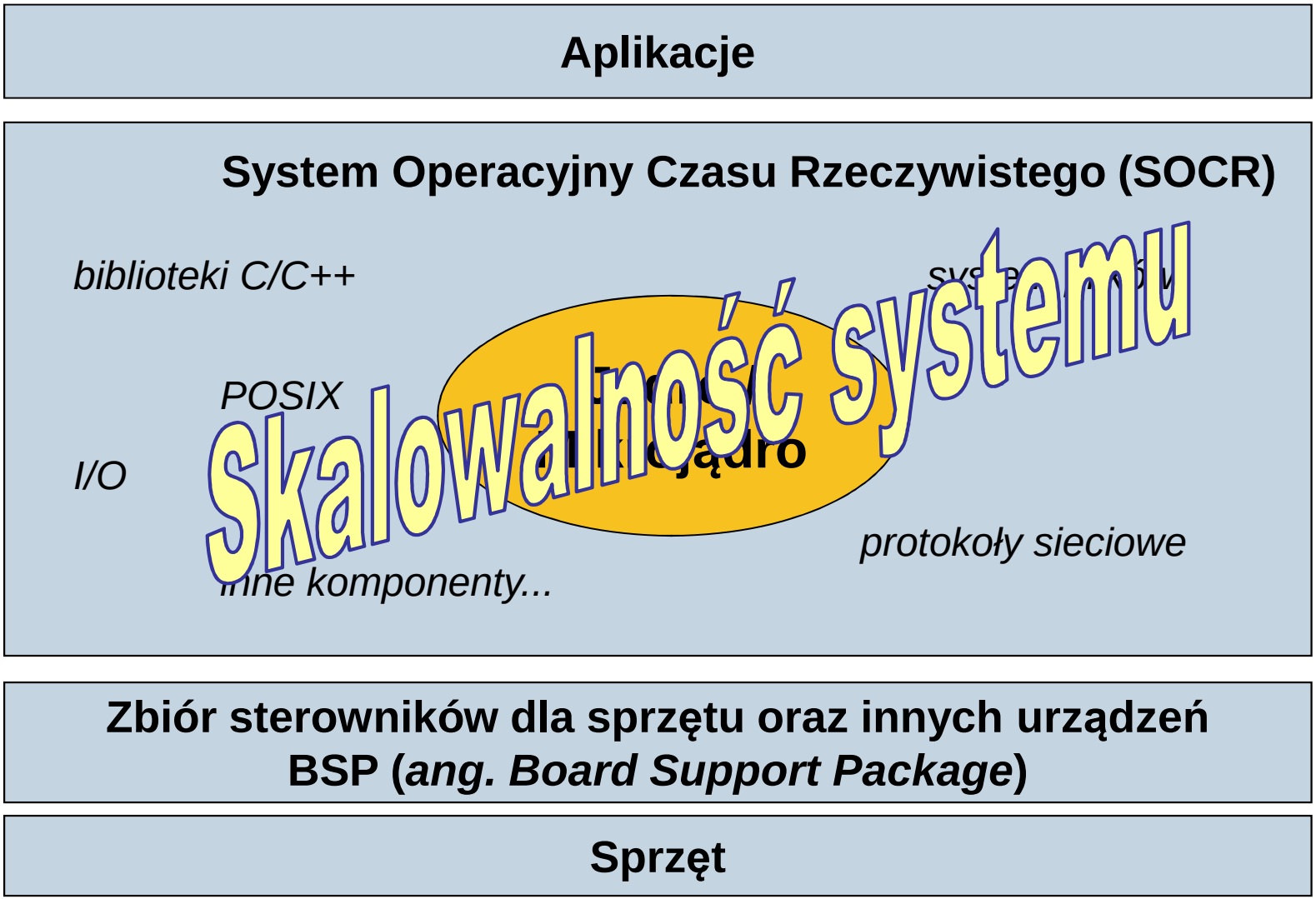
Podstawowe różnice pomiędzy SOCR a „zwykłym” SO

W systemach SOCR:

- priorytety przydzielane są do konkretnych zadań i w oparciu o nie przydzielane są zasoby
- zarządzanie zasobami jest o wiele bardziej restrykcyjne (niż w przypadku „zwykłego” SO) co powoduje skuteczne eliminowanie opóźnień
- rezygnuje się z pamięci wirtualnej (inaczej niż w przypadku „zwykłego” SO)
- mogą wystąpić niebezpieczne sytuacje (opóźnienia) wynikające z braku odpowiedniego rejestru pamięci (zasobu) w chwili gdy uruchomione programy będą potrzebowały więcej zasobów niż jest dostępnych w systemie

© SCR 2017

Ogólna struktura Systemu Operacyjnego Czasu Rzeczywistego



© SCR 2017

Podstawowe modele architektury systemów czasu rzeczywistego

- Typowa architektura systemu operacyjnego czasu rzeczywistego
  (*ang. Real-Time Executive Kernel*)
- Architektura monolityczna
  (ang. Monolithic Kernel)
- Architektura z mikrojądrem
  (*ang. Microkernel*)

© SCR 2017

Typowy model architektury SOCR

Zalety:

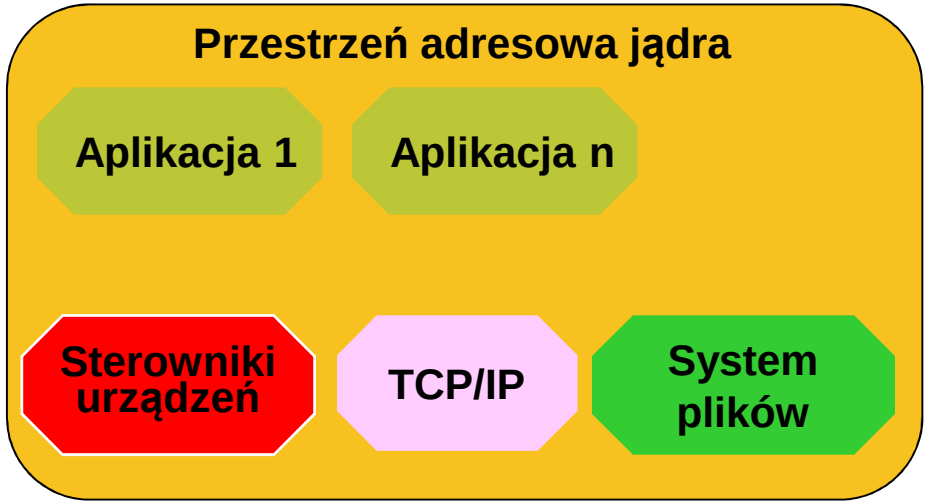
- pojedyncza przestrzeń adresowa

Wady:

- pojedyncza przestrzeń adresowa
- brak ochrony pamięci
- utrudniony rozwój i testowanie

Co w przypadku błędu?

- gdy zawiedzie aplikacja lub sterownik to konieczny restart systemu



© SCR 2017

Model architektury monolitycznej SOCR

Zalety:

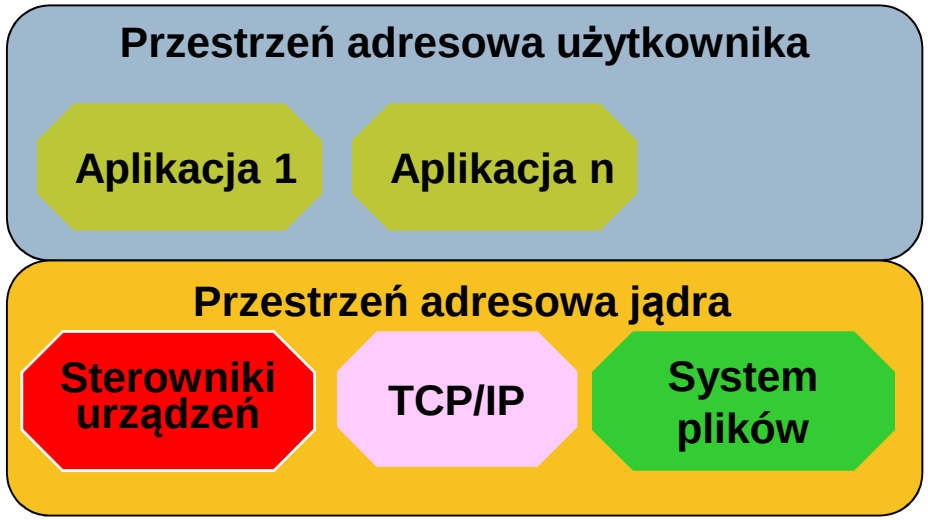
- aplikacje uruchamiają się w wydzielonej przestrzeni adresowej

Wady:

- sterowniki oraz pozostałe procesy pozostają nadal w przestrzeni adresowej jądra,
- utrudniony rozwój i testowanie

Co w przypadku błędu?

- gdy zawiedzie aplikacja system działa dalej
- gdy zawiedzie sterownik lub inny proces działający w przestrzeni adresowej jądra, może dojść do zawieszenia systemu, zatem konieczny restart systemu



© SCR 2017

Model architektury z mikrojądrem SOCR

Zalety:

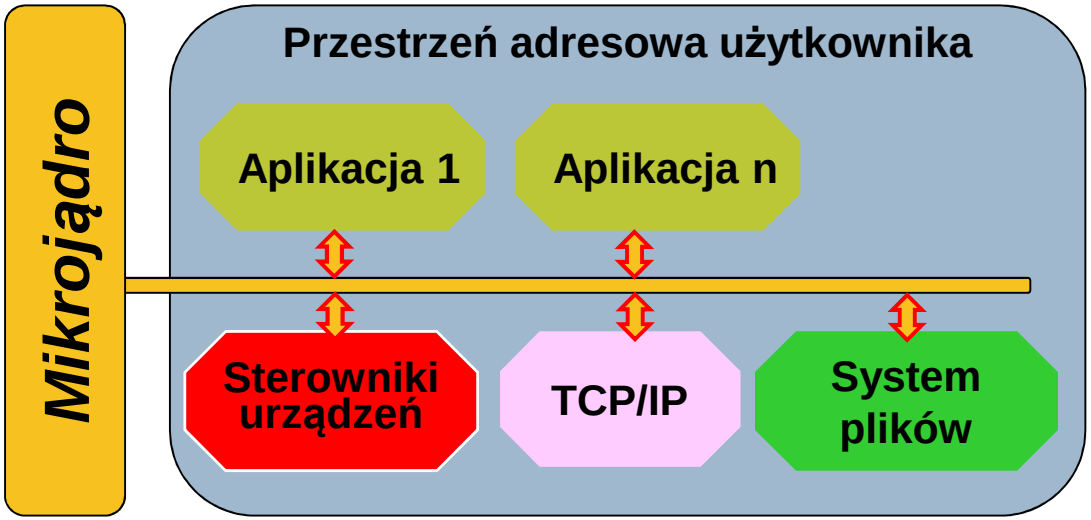
- wszystkie komponenty systemu pracują we własnych obszarach pamięci chronionej,
- dodawanie/usuwanie usług w trakcie pracy sytemu (*„w locie”*)

Wady:

- „straty czasu” związane z przełączaniem kontekstu

Co w przypadku błędu?

- gdy zawiedzie aplikacja lub sterownik nie oznacza to restartu systemu,
- system wydziedzicza „uszkodzony” proces i zwalnia zajmowaną przez niego pamięć a system działa dalej



*Jedynym **„pewnym”** modułem systemu jest mikrojądro.*
*Pozostałe elementy systemu są **„niepewne”**.*

© SCR 2017

System Operacyjny Czasu Rzeczywistego - dlaczego **warto** stosować tego typu system?

- Wykorzystanie SOCR umożliwia szybsze tworzenie programu użytkowego:
  - odciążenie programisty od stałych fragmentów programu np. inicjalizacja procesora czy organizacja pętli
  - podział programu na części i równoczesne tworzenie wielu programów z centralnym zarządzaniem pamięcią (programowanie współbieżne)
- Wykorzystanie SOCR umożliwia zwiększenie niezawodności działania programu użytkowego (wykorzystanie funkcji dostarczanych przez system operacyjny np. przydzielanie obszarów pamięci)
- Wykorzystanie SOCR umożliwia ułatwienie zarządzania procesem tworzenia oprogramowania (wykorzystanie możliwości systemu operacyjnego ułatwiającego wprowadzanie zmian, rozbudowy programu, zmiany platformy sprzętowej)

© SCR 2017

System Operacyjny Czasu Rzeczywistego - dlaczego **nie** stosować tego typu system?

- Koszty systemu operacyjnego
- Dziedzina, rodzaj zastosowania (jakie zadania będą za jego pomocą realizowane)
- Trudności z dobrym poznaniem właściwości danego SOCR przed rozpoczęciem programowania aplikacji użytkowych
- Konieczność dostosowania się do wymagań projektowania aplikacji dla SOCR

© SCR 2017

Standard POSIX

© SCR 2017

Czym jest POSIX?

POSIX (*ang. Portable Operating System Interface for Unix*):

- jest zbiorem standardów określającym jakie cechy musi spełnić aplikacja aby mogła być przenoszona pomiędzy systemami na poziomie kodu źródłowego

POSIX standaryzuje:

- interfejs programistyczny (API)
- interfejs użytkownika
- właściwości powłoki systemu

Standard POSIX - historia

Prace nad POSIX-em rozpoczęto w 1985 roku – próba standaryzacji różnych odmian systemu UNIX

Początkowo pracami kierowało IEEE (ang. Institute of Electrical and Electronics Engineers), stąd: **POSIX = IEEE 1003**

Od 1996 roku pieczę nad rozwojem standardu POSIX trzyma The Open Group (konsorcjum przemysłowe sponsorowane przez IBM, Sun, HP, Hitachi, Fujitsu, NEC) ze współpracą z IEEE

Kolejne wydania standardu POSIX noszą nazwę:

Single UNIX Specification, Version x

Od 2003 roku POSIX jest normą międzynarodową:

ISO/IEC 9945:2003

Aktualnym jest POSIX.1-2008, który jest równocześnie standardem IEEE 1003.1-2008

http://pubs.opengroup.org/onlinepubs/9699919799//

© SCR 2017

Standard POSIX – P1003

P1003.1

Definiuje interfejs aplikacji tak, aby była ona w pełni przenośna pomiędzy różnymi systemami

P1003.1a

Zestaw różnych interpretacji, wyjaśnień i rozszerzeń

P1003.1b

Rozszerzenia dotyczące systemów czasu rzeczywistego

P1003.1c

Dodanie funkcji wspierających wątki (np.: wiele wątków w ramach jednego procesu)

P1003.1d

Kolejne rozszerzenia wspierające systemy czasu rzeczywistego (np. standardowe funkcje obsługi przerwań)

P1003.1e

Rozszerzenia dotyczące bezpieczeństwa systemu spełniające kryteria bezpieczeństwa opublikowane przez Departament Obrony USA w 'Trusted Computer System Evaluation Criteria' (TCSEC)

ltd. ***P1003.1f***, ***P1003.1g***, ***P1003.1h***, ***P1003.2***, ***P1003.2b***, ***P1003.2c***, ***P1003.2d***, ***P1003.3***, ***P1003.5***, ***P1387***, ***P1003.9***, ***P1003.10***, ***P1003.11***, ***P1003.13***, ***P1003.14***, ***P1003.16***, ***P1224.2***, ***POSIX.18***, ***POSIX.19***, ***POSIX.20***, ***POSIX.21***, ***POSIX.0***

© SCR 2017

Standard POSIX – P1003.1b, P1003.1c, P1003.1d

- semafory
- proces blokowania pamięci
- pliki mapowane w pamięci i współdzielona pamięć
- kolejkowanie priorytetowe
- obsługa sygnałów w czasie rzeczywistym
- liczniki czasu
- komunikacja międzyprocesowa
- synchroniczne operacje wejścia/wyjścia (I/O)
- asynchroniczne operacje wejścia/wyjścia (I/O)

© SCR 2017

Standard POSIX – P1003.1b, P1003.1c, P1003.1d

- funkcje wspierające wielowątkowość w ramach jednego procesu
- funkcje kontrolujące wątki i ich atrybuty
- muteksy (specjalnego rodzaju semafory binarne),
- zmienne warunkowe
- funkcje umożliwiające monitorowanie czasu wykonywania procesów i wątków oraz określające limity czasów oczekiwania dla funkcji blokujących
- normy dotyczące rozproszonej komunikacji czasu rzeczywistego, obsługi urządzeń z buforami oraz wysyłania bloków kontrolnych
- usługi dla niezawodności, dostępności oraz użyteczności

© SCR 2017

Standard POSIX – podsumowanie

POSIX daje pewną dowolność twórcom systemów czasu rzeczywistego, nie muszą oni implementować całości, a tylko wybrane przez siebie funkcje opisane w standardzie.

UWAGA: Wymagane jest jasne określenie co jest, a czego nie ma w systemie – odnośnie implementowanych funkcji ze standardu POSIX.

W praktyce, zgodność systemu ze standardem 1003.1b gwarantuje poprawne wykonywanie zadań czasu rzeczywistego o charakterze miękkim.

© SCR 2017

Podstawowe Pojęcia

© SCR 2017

Podstawowe pojęcia – Program, Zadanie

Program

- Program jest zapisem (implementacją) algorytmu, realizującego założony cel poprzez operowanie na pewnych strukturach danych

PROGRAM = ALGORYTM + STRUKTURY DANYCH

Zadanie

- Zadanie jest wykonującym się programem
- Składa się z „małych” niezależnych procesów, które sądzą „że mają procesor na swoją wyłączność”
- Poszczególne zadania konkurują między sobą o czas procesora

© SCR 2017

Podstawowe pojęcia – Proces, Wątek

Proces

- Proces jest składnikiem zadania
- Proces składa się z wątków
- Każdy uruchomione w systemie proces zawiera szereg charakterystycznych dla niego parametrów, takich jak:
  - unikalny numer
  - priorytet
  - kontekst (stan zestawu rejestrów procesora)
  - obszar stosu
  - kod programu

Wątek

- Podstawowa jednostka wykorzystania procesora
- Grupa wątków (składających się na proces) dzieli kod, przestrzeń adresową i zasoby systemu operacyjnego

© SCR 2017

Podstawowe pojęcia - Zasób

Zasób

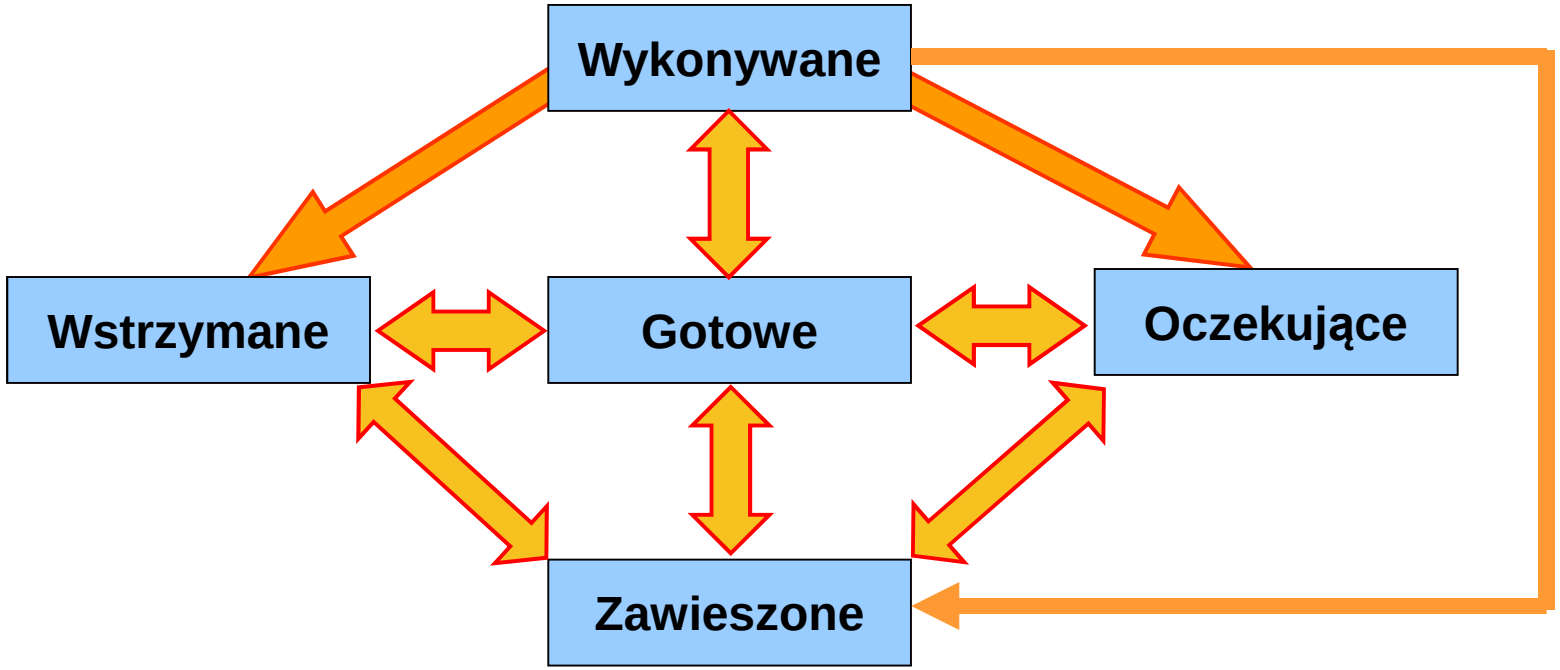
- Zasób jest elementem systemu wykorzystywanym przez zadania
- Zasobem może być:
  - urządzenie I/O (klawiatura, wyświetlacz)
  - pamięć (zmienna, stała, macierz itp.)
  - rejestr
  - interfejs
  - czas procesora
  - stos

Zasób współdzielony

- Zasób, który może być wykorzystywany przez więcej niż jedno zadanie
- Każde zadanie powinno uzyskiwać wyłączny dostęp do współdzielonego zasobu, aby uniknąć przekłamania danych - technika **wzajemnego wykluczania** (*ang. Mutual Exclusion*)

© SCR 2017

Podstawowe pojęcia – Zadanie (cd.)-
- przykład możliwych zmian stanu zadnia



Stany zadania – różne w różnych systemach

© SCR 2017

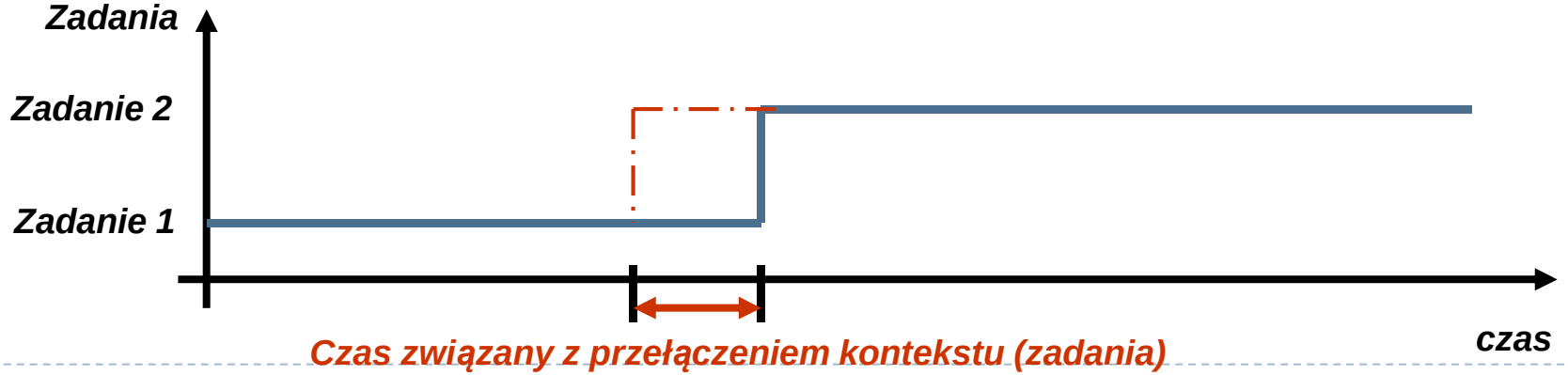
Podstawowe pojęcia – Zadanie (cd.)- - przykład możliwych zmian stanu zadnia

- Nowo utworzone zadanie lub zadanie które nie może być wykonane otrzymuje status **„Zawieszone”**
- Po procesie aktywacji zadanie przechodzi do stanu **„Gotowe”** i od tej chwili podlega szeregowaniu zadań przez jadro (odpowiedni algorytm szeregowania)
- Zadanie o statusie **„Wykonywane”** posiada procesor
- Zadanie które nie czeka na inny zasób niż procesor jest w stanie **„Gotowe”**
- Jeżeli zadanie oczekuje na jakiś zasób to jest w stanie **„Wstrzymane”**
- Zadanie jest w stanie **„Oczekujące”** jeżeli czeka na upływ pewnego przedziału czasu
- Zadanie może być usunięte w każdym stanie przez jądro systemu

© SCR 2017

Przełączanie kontekstu (zadania)

- Kiedy jądro systemu decyduje o uruchomieniu innego zadania, kontekst obecnego zadania (rejestry CPU) zostaje zapisany w przeznaczonym dla danego zadania obszarze pamięci
- Następnie kontekst nowego zadania zostaje odczytany z odpowiedniego obszaru i zostaje wznowione wykonanie nowego zadania
- Proces ten nosi nazwę przełączania kontekstu lub zadania
- Przełączanie zadań stanowi martwy czas programu



© SCR 2017

Priorytety Statyczne

- Priorytety zadań są priorytetami **statycznymi**, wtedy gdy priorytet każdego z zadań **nie zmienia** się podczas wykonania aplikacji
- Każde zadanie ma więc przydzielony ustalony priorytet (na podstawie znanych wymagań i ograniczeń czasowych zadania) podczas kompilacji

Priorytety Dynamiczne

- Priorytety zadań są priorytetami **dynamicznymi** jeśli priorytety zadań **mogą być zmienianie** podczas wykonania, uruchamiania aplikacji

Priorytety dynamiczne to pożądana właściwość systemów SOCR dla uniknięcia *inwersji priorytetów (...o tym dalej)*.

© SCR 2017

Jądro systemu

Jądro jest częścią systemu wielozadaniowego odpowiedzialną za:

- przełączanie kontekstu (zadania)
- **zarządzanie zadaniami** (a więc zarządzanie czasem procesora)
- oraz **komunikację pomiędzy zadaniami**

Wykorzystanie jądra czasu rzeczywistego upraszcza projektowanie systemów
(np. sterowania, monitorowania)
dzięki umożliwieniu podziału problemu na wiele zadań zarządzanych przez jądro
(jednocześnie lub współbieżnie)

© SCR 2017

Typy jąder opartych na priorytetach

Typy jąder systemu SOCR oparte na priorytetach:

- jądro **bez wywłaszczania** (*ang. non-preemptive*)
- jądro **z wywłaszczaniem** (*ang. preemptive*)

© SCR 2017

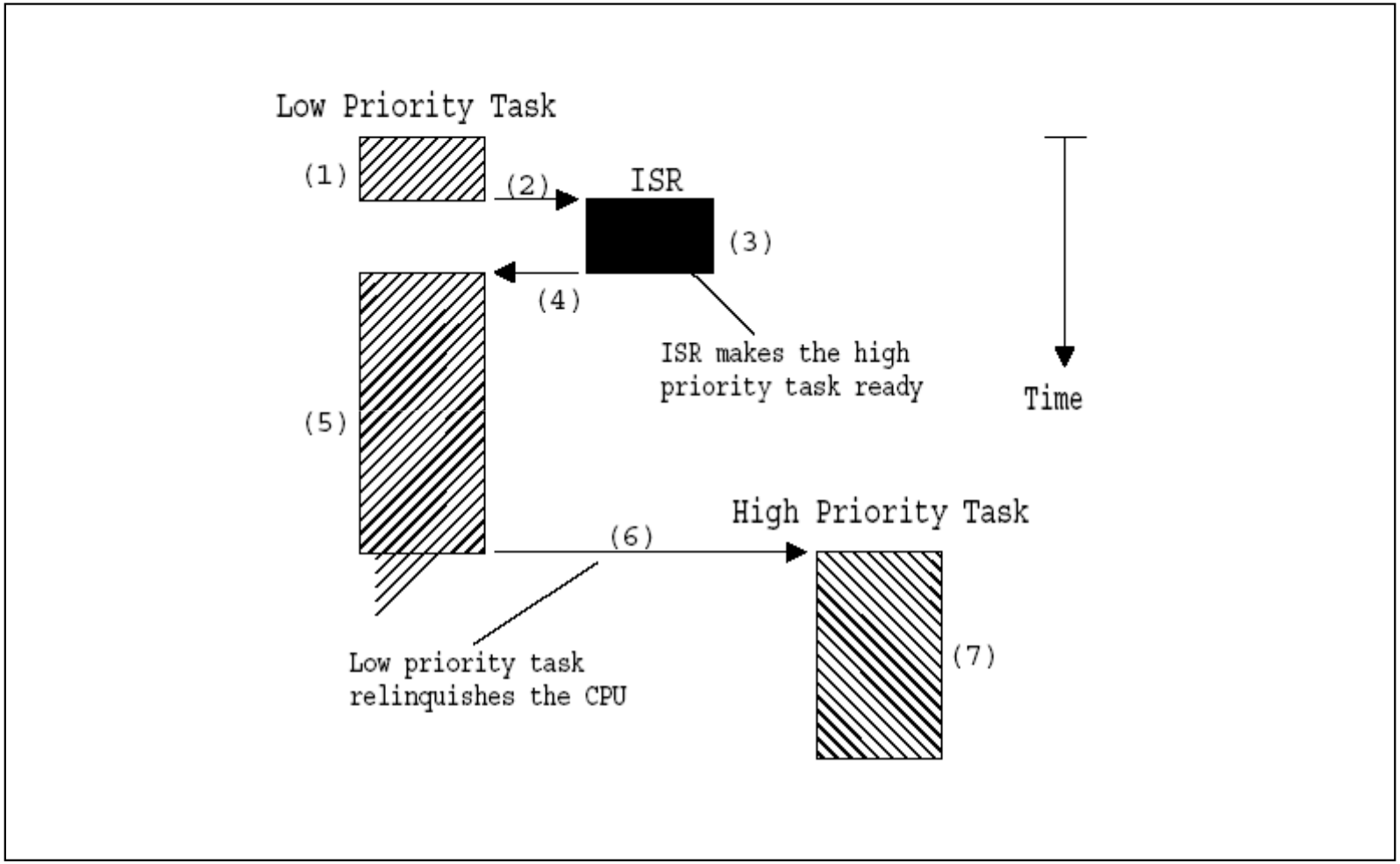
Jądro bez wywłaszczania

- W tym typie jądra wymagane jest aby każde zadanie samodzielnie oddało kontrolę nad procesorem
- Zachodzące zdarzenia obsługiwane są przez system obsługi przerwań ISR (*ang. Interput Service Routines*)
- Procedura obsługi przerwania może ustawić zadanie o wyższym priorytecie jako gotowe, ale i tak następuje powrót do aktualnego zadania
- Zadanie o wyższym priorytecie uzyska kontrolę nad procesorem, gdy zadanie o niższym priorytecie ją odda

Niewiele systemów komercyjnych jest wykonywanych
bez mechanizmu wywłaszczania.
np. Microsoft Windows 3.11, Mac OS 9 (i wcześniejsze)

© SCR 2017

Jądro bez wywłaszczania



ISR – procedury obsługi przerwań

© SCR 2017

Jądro z wywłaszczeniem

- Zadanie o najwyższym priorytecie gotowe do uruchomienia, zawsze otrzymuje kontrolę nad procesorem
- Zachodzące zdarzenia obsługiwane są przez system obsługo przerwań ISR
- W systemach z wywłaszczaniem czas wykonania zadania o najwyższym priorytecie jest określony, można dokładnie określić, kiedy zadanie o najwyższym priorytecie uzyska kontrolę nad procesorem
- Przerwanie wywłaszcza aktualnie wykonywane zadanie, a po zakończeniu jego obsługi, jądro systemu rozpoczyna wykonanie zadania o najwyższym priorytecie, które jest gotowe, a nie przerwane zadanie

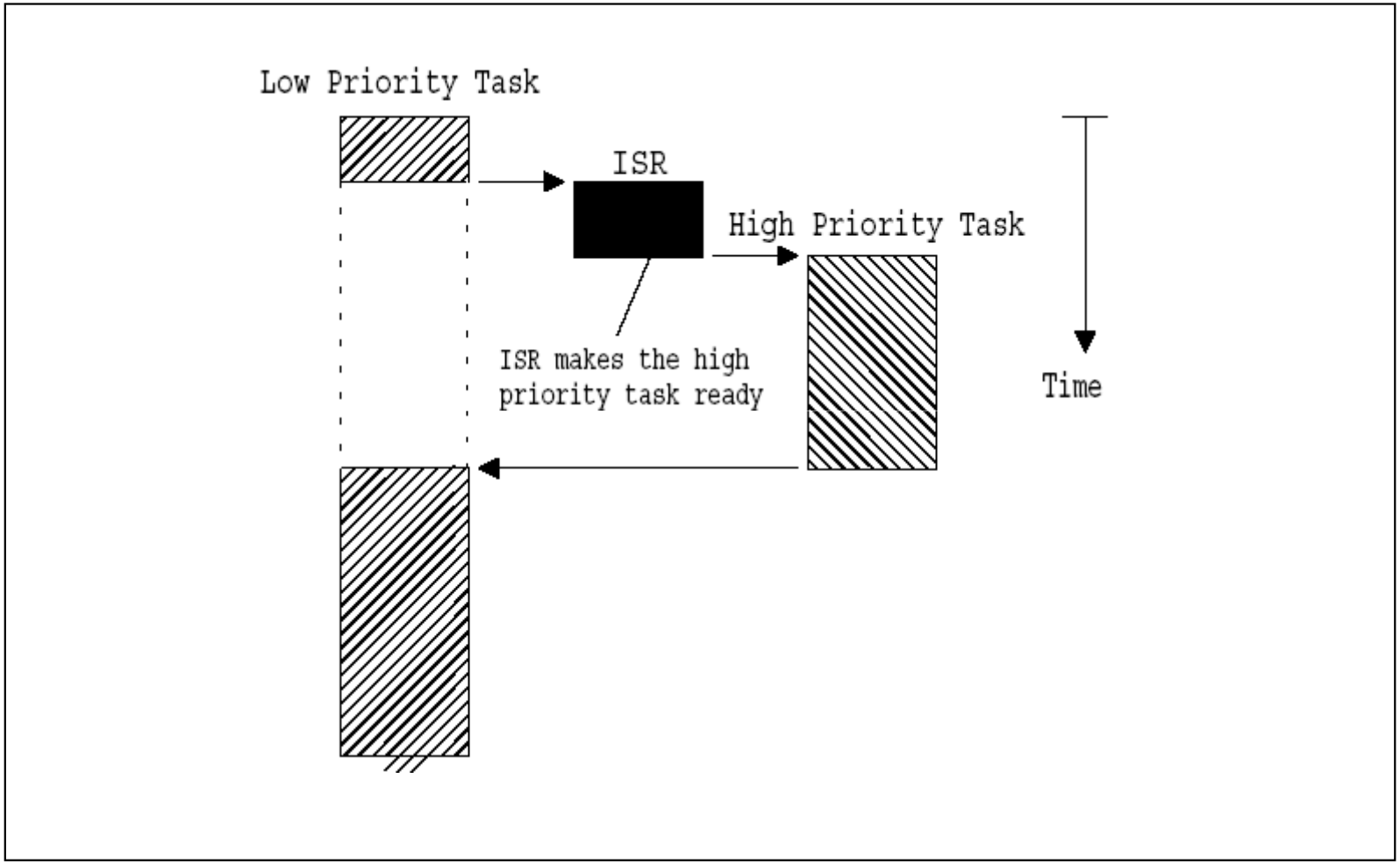
*Systemy **z wywłaszczaniem** są stosowane gdy **przewidywalność** systemu **jest ważna***

Większość komercyjnych systemów jest systemami z wywłaszczaniem

np. przeważająca większość systemów UNIXowych, Msc OS X, Microsoft Windows (95, 98, ME, NT, 2000, XP, Vista,…)

© SCR 2017

Jądro z wywłaszczaniem



ISR – procedury obsługi przerwań

© SCR 2017

Bibliografia:

[1] P.Szymczyk (2003). Systemy Operacyjne Czasu Rzeczywistego. Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Kraków.

[2] J.Ułasiewicz (2007). System czasu Rzeczywistego QNX6 Neutrino. Wydawnictwo BTC, Legionowo.

[3] K.Lal, T.Rak, K.Orkisz (2003). RTLinux system czasu rzeczywistego. Helion, Gliwice.

© SCR 2017

Dziękuję za uwagę !!!

© SCR 2017