Imię i nazwisko: _____ Numer indeksu: _____

# Systemy operacyjne

*6 lutego 2019*

*czas trwania: 180 minut*

| Punkty | 0 – 49 | 50 – 59 | 60 – 69 | 70 – 79 | 80 – 89 | 90 – 100 |
|---|---|---|---|---|---|---|
| Ocena | 2.0 | 3.0 | 3.5 | 4.0 | 4.5 | 5.0 |

**Uwagi do zadań 1–16:** Zdania prawdziwe oznacz literą T, a fałszywe literą N. Funkcja $p$, która liczbie poprawnie udzielonych odpowiedzi przyporządkowuje liczbę punktów za zadanie, jest określona następująco: $p(4) = 3$, $p(3) = 2$, $p(2) = 1$, $p(1) = 0$, $p(0) = 0$.

**Zadanie 1 (3).** Architektura systemu operacyjnego.

T Awaria sterownika w jądrze monolitycznym może doprowadzić do załamania się systemu.

N Jądro monolityczne nie umożliwia doładowywania sterowników w trakcie działania systemu.

N Mikrojądro zawiera w sobie algorytmy szeregowania wątków.

N Jądro jest wywłaszczalne, jeśli przerwania mogą się zagnieżdżać.

**Zadanie 2 (3).** O procesach ogólnie.

T Obsługa wywołania systemowego wymaga zmiany trybu pracy procesora.

N Jądro może zmienić stan procesu z `READY` na `BLOCKED`.

N Rekord procesu przechowuje kontekst procesora.

T Mechanizm śledzenia procesów pozwala jednemu procesowi czytać komórki pamięci należące do innego procesu.

**Zadanie 3 (3).** O wątkach ogólnie.

N Przełączanie kontekstu między wątkami należącymi do dwóch różnych procesów jest równie szybkie co przełączanie wątków należących do tego samego procesu.

N Mikrowątki (włókna) nie posiadają własnego stosu.

N Wątki użytkownika (ang. *ULT*) mogą się synchronizować z użyciem muteksów udostępnionych przez jądro.

N Lokalna przestrzeń wątku (ang. *TLS*) jest niedostępna dla innych wątków w danym procesie.

**Zadanie 4 (3).** Procesy i wątki w **systemach uniksowych**.

N Proces można zawsze zakończyć poprzez wysłanie sygnału `SIGTERM`.

N Sierota to proces pozbawiony wszystkich zasobów z wyjątkiem rekordu procesu.

T Proces, który przeszedł w płytki (ograniczony czasowo) sen, można wybudzić sygnałem.

N Proces zawsze posiada tożsamość użytkownika, który go utworzył.

**Zadanie 5 (3).** Programowanie z użyciem procesów **systemu uniksowego**.

T Proces utworzony wywołaniem `fork` dziedziczy po swoim rodzicu otwarte połączenia sieciowe.

T Żeby uruchomić nowy proces ładowany z pliku należy użyć pary wywołań systemowych `fork` i `execve`.

N Jeśli nastąpi powrót z procedury, od której zaczęło się wykonanie procesu, to proces ten zakończy się poprawnie.

N Zmienne środowiskowe są przechowywane w pamięci współdzielonej między spokrewnionymi ze sobą procesami.

**Zadanie 6 (3).** O synchronizacji ogólnie.

T Jeśli dużo wątków często synchronizuje się na małej liczbie blokad, to zachodzi rywalizacja o blokady.

T Implementacja zmiennej warunkowej nie przechowuje wartości logicznej sprawdzanego predykatu.

N Blokada wirująca to wydajna implementacja sekcji krytycznej w systemach jednoprocesorowych.

N Wątek jest głodzony, jeśli postępuje z obliczeniami, ale system ciągle odrzuca jego żądania o dostęp do zasobu.

**Zadanie 7 (3).** O komunikacji ogólnie.

N Potoki umożliwiają jedno- i dwukierunkową komunikację między procesami na różnych maszynach.

N Transmisja z użyciem gniazd strumieniowych dzieli dane na paczki zwane datagramami.

T Używając protokołu bezpołączeniowego przy każdym odbiorze pakietu dostajemy adres nadawcy.

N Zdalne wywołanie procedur dopuszcza przekazywanie listy otwartych połączeń jako argumentu funkcji.

**Zadanie 8 (3).** Synchronizacja i komunikacja w **systemach uniksowych**.

T `POSIX mutex` musi być blokowany i zwalniany przez ten sam proces lub wątek.

N Wątek, który oczekiwał na zmiennej warunkowej, po wybudzeniu wchodzi jako pierwszy do monitora.

T Pamięć współdzieloną między procesami można zaimplementować z użyciem wywołania `mmap`.

N Przesyłanie komunikatów z użyciem potoków jest zawsze atomowe — nie występuje fragmentacja danych.

**Zadanie 9 (3).** O szeregowaniu zadań ogólnie.

N Dyspozytor jest algorytmem wybierającym następne zadanie do wykonania.

N W szeregowaniu z priorytetami nie można wywłaszczyć zadania przed upływem przydzielonego mu kwantu czasu.

T Szeregowanie loteryjne uwzględnia priorytety przyznając ważniejszym zadaniom więcej losów.

N Problem odwrócenia priorytetów rozwiązuje się poprzez tymczasowe podwyższenie priorytetu wątku, który oczekuje na zasób zablokowany przez wątek o niższym priorytecie.

**Zadanie 10 (3).** Algorytmy przydziału pamięci.

N Problem fragmentacji wewnętrznej można rozwiązać dzięki kompaktowaniu.

N Strategia *best–fit* w ogólnym przypadku minimalizuje fragmentację zewnętrzną.

N Znaczniki graniczne (ang. *boundary tag*) służą do szybkiego znajdowania następnego bloku pamięci.

T Leniwe złączanie nieużytków jest korzystne w przypadku przydziału pamięci dla krótko żyjących bloków.

Imię i nazwisko: _____ Numer indeksu: _____

**Zadanie 11 (3).** Pamięć wirtualna.

T Odwzorowanie plików w pamięć jest używane do współdzielenia bibliotek konsolidowanych dynamicznie.

T Stronicowanie na żądanie podczepia do przestrzeni adresowej tylko te ramki, do których wykonano dostęp.

N Jądro wspierające kopiowanie przy zapisie używa mniej pamięci wirtualnej, niż system bez tej funkcjonalności.

N Główny błąd strony jest generowany przy dostępie do nieprzydzielonego obszaru adresów wirtualnych.

**Zadanie 12 (3).** Zarządzanie pamięcią wirtualną.

T Ta sama ramka może być podpięta do wielu wirtualnych przestrzeni adresowych.

N Gdy zbiór rezydentny jednego procesu jest mniejszy od jego zbioru roboczego, to system się szamocze.

N Polityka ładowania ustala adresy fizyczne, pod którymi umieszcza się strony wczytywanego procesu.

T Demon stronicowania odpowiada za uspójnianie brudnych stron z pamięcią drugorzędną.

**Zadanie 13 (3).** Pliki w **systemach uniksowych**.

T Istnieją takie pliki, których zawartość nie jest przechowywana w pamięci drugorzędnej.

N Ścieżka bezwzględna, to najkrótsza ścieżka od korzenia do pliku niezawierająca dowiązań symbolicznych.

T System plików Uniksa nie rozróżnia plików typu binarnego i tekstowego.

N Pusty katalog można usunąć za pomocą wywołania systemowego «`unlink`».

**Zadanie 14 (3).** Przydział przestrzeni dyskowej i systemy plików.

N I–węzeł przechowuje nazwę i typ pliku.

T Leniwy przydział bloków zapobiega powstawaniu fragmentacji w trakcie tworzenia szybko rosnących plików.

N Naprawa bitmapy wolnych bloków wymaga przeskanowania wszystkich bloków systemu plików.

T Odczytanie i–węzła małego pliku umożliwia wyznaczenie numerów wszystkich bloków bez kolejnych operacji I/O.

**Zadanie 15 (3).** Szeregowanie w systemach wieloprocesorowych i czasu rzeczywistego.

T Szeregowanie według powinowactwa stara się uruchamiać wątki z tego samego procesu jeden po drugim.

T Szeregowanie zespołowe jest strategią szeregowania wątków, które często się ze sobą komunikują.

T Przyciąganie i wypychanie zadań nie rozwiązuje problemu równomiernego obciążenia procesorów.

N Opóźnienie ekspedycji zawiera w sobie czas opóźnienia obsługi przerwania.

**Zadanie 16 (3).** O wirtualizacji ogólnie.

T Monitor maszyn wirtualnych typu 2 działa pod kontrolą systemu operacyjnego gospodarza.

T Jądro systemu gościa może użyć parawirtualizacji do zastąpienia instrukcji wrażliwych.

T Balonikowanie jest metodą odbierania systemowi gościa zwirtualizowanej pamięci fizycznej.

T Monitor może zastąpić urządzenie I/O używając pamięci współdzielonej i zestawu hiperwywołań.

**Zadanie 17 (6).** Rozwiń anglojęzyczne skróty często pojawiające się w kontekście systemów operacyjnych:

| | | | |
|---|---|---|---|
| IPC | Inter-Process Communication | HAL | Hardware Abstraction Layer |
| UID | User Identifier | VM | Virtual Memory / Machine |
| PCB | Process Control Block | FS | File System |
| RPC | Remote Procedure Call | COW | Copy on Write |

**Zadanie 18 (10).** Bariera to narzędzie synchronizacyjne służące do synchronizacji grup $k$ wątków. Po zatrzymaniu się co najmniej $k$ wątków na barierze (procedura «wait») w jednym kroku opuszcza ją dokładnie $k$ wątków. Wątki przychodzą do bariery i opuszczają ją w tej samej kolejności. Po jednokrotnym użyciu bariera wraca do stanu nominalnego. Sumaryczna liczba wątków korzystających z bariery może być dużo większa niż $k$. Poniższa implementacja zawiera co najmniej jeden błąd. Wskaż stan, w którym funkcja «wait» zadziała nieprawidłowo.

```
1  fun barrier@init(int k):
2    turnstile1 = new Semaphore(0)
3    turnstile2 = new Semaphore(0)
4    mutex = new Semaphore(1)
5    count = 0
6    n = k
7
8  fun barrier@wait():
9    mutex.wait()
10     count += 1
11     if count == n:
12       turnstile1.signal(n)
13   mutex.signal()
14   turnstile1.wait()
15   mutex.wait()
16     count -= 1
17     if count == 0:
18       turnstile2.signal(n)
19   mutex.signal()
20   turnstile2.wait()
```

Weźmy $k+1$ wątków $t_i$ numerowanych od 1. W tabelkę poniżej wpisz sekwencję zdarzeń, która prowadzi do osiągnięcia nieprawidłowego stanu.

| wątek | po wierszu | stan |
|---|---|---|
| $t_1$ | 8 | aktywny |
| $t_1 \ldots t_{k+1}$ | 13 | aktywny |
| $t_1 \ldots t_k$ | 14 | aktywny |
| $t_{k+1}$ | 14 | zablokowany |
| $t_1 \ldots t_k$ | 20 | zablokowany |

**UWAGA!** Stan wątku należy zapisać następująco: zablokowany, aktywny.

W stanie niedozwolonym wartość semafora «turnstile1» wynosi -1, «turnstile2» wynosi -k a wartość zmiennej «count» jest równa 1. Poniżej należy słownie scharakteryzować nieprawidłowy stan:

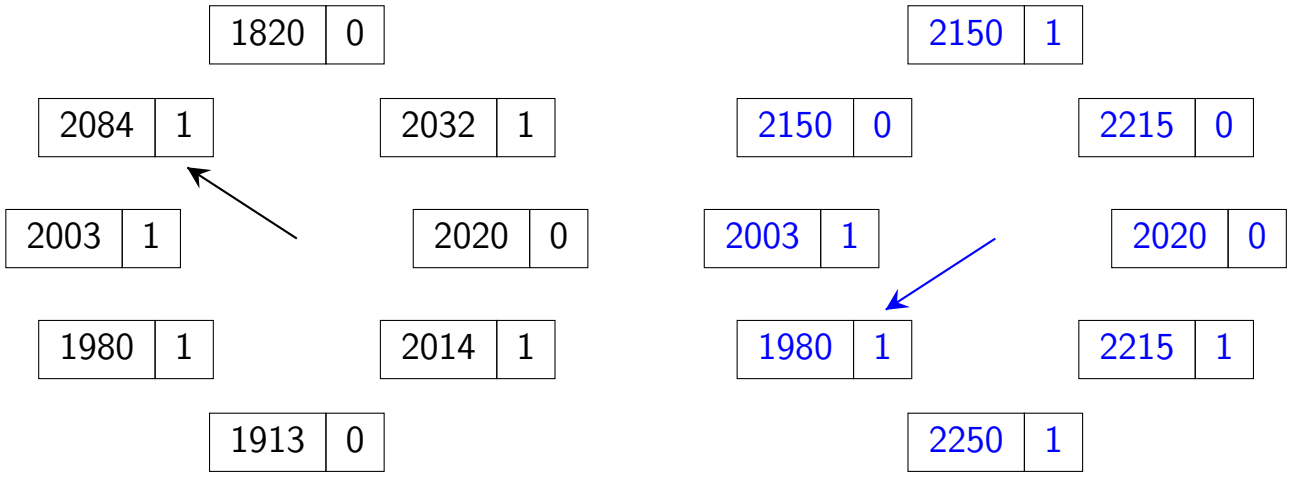
$t_1 \ldots t_k$ powinny opuścić barierę, ale nigdy nie tego zrobią, nawet jeśli przyjdą nowe wątki $t_i$, gdzie $i > k+1$.

**UWAGA!** Ujemna wartość semafora oznacza liczbę wątków oczekujących na semaforze.

Imię i nazwisko: _____ Numer indeksu: _____

**Zadanie 19 (8).** Na rysunku po lewej widzimy stan algorytmu WSClock z chwili $t_0 = 2100$. Każdej ramce przypisano parę wartości na użytek algorytmu. Po lewej widnieje liczba oznaczająca ostatni czas, w którym algorytm oznaczył stronę jako używaną. Po prawej wartość bitu «referenced», którą zarejestrowano od ostatniego przebiegu algorytmu przez tą stronę. Stała $\tau$ wyznaczająca okno czasowe dla stron ze zbioru rezydentnego wynosi 200. Ramię zegara wskazuje na następną stronę do przetworzenia.

Na rysunku po prawej stronie narysuj stan końcowy algorytmu po przetworzeniu żądań zastąpienia strony w chwilach $t_1 = 2150$, $t_2 = 2215$, $t_3 = 2250$. Należy założyć, że sprzęt nie aktualizuje w międzyczasie wartości bitów «referenced». Nie zapomnij narysować końcowej pozycji ramienia zegara.




**Zadanie 20 (10).** Rozważmy system plików podobny do Ext2. Interesuje nas usunięcie dużego pliku z katalogu zajmującego dwa bloki. Zachodzi sytuacja, w której należy przeprowadzić kompaktowanie katalogu. Uszereguj akcje na systemie plików, tak by nie dopuścić do utraty spójności metadanych. Wiemy, że w razie awarii systemu program «fsck» poradzi sobie ze zwolnieniem nieosiągalnych danych lub naprawieniem licznika referencji i-węzłów.

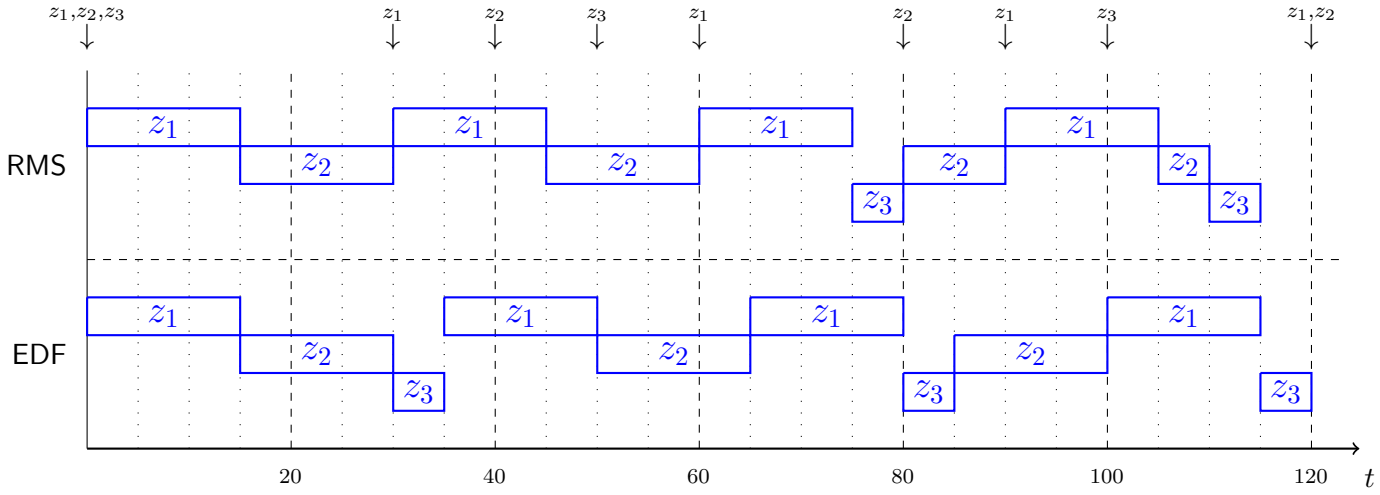
Bitmapa wolnych bloków / i-węzłów zajmuje tylko jeden blok, a wpis katalogu nigdy nie leży na granicy dwóch bloków. Każda akcja wpisana w pole poniżej musi dotyczyć co najwyżej jednego bloku albo i-węzła. Może składać się z jednego odczytu, modyfikacji danych w pamięci lub zapisu na dysk. Można używać sformułowań „dla każdego”.

1. Wczytaj blok katalogu $d_1$ zawierający wpis, usuń wpis.
2. Wczytaj drugi blok katalogu $d_2$.
3. Wczytaj bitmapę bloków, zarezerwuj blok $b$, zapisz bitmapę.
4. Do bloku $b$ wpisz połączoną zawartość $d_1$ i $d_2$.
5. Zaktualizuj i-węzeł katalogu, zastępując wskaźniki na $d_1$ i $d_2$ wskaźnikiem na $b$, oraz zmniejszając liczbę referencji i czas ostatniej modyfikacji.
6. Zwolnij blok $d_1$ i $d_2$, zapisz bitmapę bloków.
7. Przeczytaj i-węzeł pliku. Jeśli ma więcej niż jedno dowiązanie, to koniec.
8. Dla każdego numeru bloku odczytanego z użyciem bloków pośrednich zwolnij blok i zapisz bitmapę bloków.
9. Wczytaj bitmapę i-węzłów, zwolnij i-węzeł, zapisz bitmapę.

**Wskazówka:** Rozwiązanie autora zadania liczy 9 punktów.

**Zadanie 21 (8).** Niech $t_i$ oznacza czas wykonania zadania, a $p_i$ okres zadania. Każde zadanie musi się zakończyć przed końcem okresu. Mamy trzy zadania czasu rzeczywistego $z_1$, $z_2$ i $z_3$, o nastepującej charakterystyce: $t_1 = 15$ i $p_1 = 30$, $t_2 = 15$ i $p_2 = 40$, $t_3 = 5$ i $p_3 = 50$. Dla algorytmu $\mathrm{RMS}$ (ang. *Rate Monotonic Scheduling*) zestaw zadań jest szeregowany wtw. gdy $\sum_{i=1}^{n} \frac{t_i}{p_i} \leq$ $n(2^{1/n} - 1)$. Zatem zestaw zadań nie jest szeregowalny ponieważ dla naszych zadań zachodzi nierówność $0.975 = 0.5 + 0.375 + 0.1 > 3(\sqrt[3]{2} - 1) \approx 0.78$.

Na poniższym diagramie narysuj rezultat szeregowania $\mathrm{RMS}$ i $\mathrm{EDF}$ (ang. *Earliest Deadline First*) dla pierwszych 120 jednostek czasu działania algorytmów. Jeśli którykolwiek z algorytmów przekorczy termin, to przerwij jego działanie.




**Wskazówka:** $\sqrt[3]{2} \approx 1.26$.

**Zadanie 22 (10).** Opisz strukturę danych wykorzystywaną przez jądro systemu uniksopodobnego do zarządzania przestrzenią wirtualną procesu i algorytm obsługi błędnego dostępu do pamięci. Jak algorytm odróżnia błąd strony od błędu programisty. Jakich danych wymaga ta procedura?

W PCB proces przechowuje listę segmentów, z których każdy zawiera pierwszy i ostatni adres wirtualny, uprawnienia dostępu `rwx` i wskaźnik na obiekt wspierający. Obiektem wspierającym może być pamięć anonimowa lub plik dyskowy.

Algorytm obsługi błędu strony dysponuje adresem wirtualnym, który spowodował wyjątek procesora, oraz rodzaj dostępu: odczyt/zapis/wykonanie. Jeśli adres nie należy do żadnego segmentu to błąd programisty, koniec. Jeśli strony nie ma w pamięci, to musimy przydzielić ramkę i wyzerować ją (pamięć anonimowa) lub załadować z dysku (odwzorowanie pliku w pamięć), koniec. Jeśli uprawnienia dostępu nie zgadzają się z uprawnieniami segmentu to błąd programisty, koniec. Jeśli segment jest `read-write`, ale strona jest `read-only`, to należy obsłużyć kopiowanie przy zapisie, tj. skopiować ramkę i podczepić w zadane miejsce z uprawnieniami do zapisu, koniec.

**Użyj pojęć:** ramka, segment, uprawnienia dostępu, obiekt wspierający, pamięć anonimowa, odwzorowanie pliku, kopiowanie przy zapisie.