Systemy Operacyjne
Programy Symulacyjne

1 Programy Symulacyjne – Wymagania

1.1 Zadania do wykonania

1. Symulacja działania wybranych algorytmów planowania czasu procesora dla zamkniętej puli zadań. Należy wybrać co najmniej dwa z poniższych algorytmów:
   (a) FCFS,
   (b) LCSF,
   (c) SJF,
   (d) Round-Robin.
2. Symulacja działania algorytmów zastępowania stron. Proszę wybrać dwa z poniższych:
   (a) FIFO,
   (b) LRU,
   (c) LFU,
   (d) MFU.

1.2 Produkt finalny

Przed prezentacją programów symulacyjnych należy oddać sprawozdanie (w wersji elektronicznej, wszystkie pliki w jednym archiwum), zawierające:

1. Kod źródłowy programu:
   - Kod powinien być zdokumentowany.
2. Dane testowe:
   - Pliki z danymi testowymi w dowolnym formacie.
3. Wyniki eksperymentów
   - Pliki z surowymi wynikami eksperymentów.
4. Sprawozdanie:
   - Opis procedury testowania algorytmów.

- Opracowane wyniki Eksperymentów.
- Wnioski.

Wszystkie pliki powinny być umieszczone w archiwum nazwanym według wzoru:

- symulacje_ < Nazwisko > _ < numer_indeksu > .$\{zip|rar\}$
  - przykładowo: symulacje_Kowalski_12345678.zip

2 Ocena

2.1 Zasady oceniania

1. Projekt musi być oddany w terminie określonym przez prowadzącego. Za spóźnienie naliczane są kary – patrz wprowadzenie.
2. Każdy student przedstawia swoją pracę prowadzącemu indywidualnie.
3. Wymagana jest znajomość struktury programu oraz funkcjonalności aplikacji.
   - Punkty za poszczególne elementy projektu **nie** zostaną przyznane jeżeli student nie zna sposobu ich działania.
   - Punkty za poszczególne elementy projektu **nie** zostaną przyznane jeżeli student nie potrafi, w obecności prowadzącego, dokonać niewielkich modyfikacji wskazanych fragmentów.

2.2 Kryteria

Oceniane będą:

- Poprawność zaimplementowania algorytmów.
- Kompletność rozwiązania (czy zostały zaimplementowane wszystkie wymagane algorytmy).
- Jakość wykonania:
  - Kod nie powinien zawierać błędów uniemożliwiających kompilację (programy niekompilujące się nie będą oceniane).
  - Program nie może zawierać błędów powodujących zakończenie się błędem wykonania.
    - Zaimplementowane rozwiązania nie powinny zawierać błędów takich jak wycieki pamięci, zapis/odczyt poza przydzieloną pamięcią itp.
  - Kod powinien być podzielony na moduły (jeden plik zawiera jeden moduł).
  - Czytelność kodu (komentarze, formatowanie). Dobry przykład jak **nie** należy formatować kodu można znaleźć tutaj [1].
- Pomysłowość zaimplementowanego rozwiązania.

- Terminowość oddawania zadań.
- Poprawność sprawozdania
  - Opis procedury testowania.
  - Opracowane wyniki eksperymentów.
  - Poprawność wniosków.

3 Literatura

[1] The international obfuscated c code contest, `http://www.ioccc.org/years.html`.