Systemy operacyjne 2

Wymagania projektowe

dr inż. Jarosław Rudy

1 marca 2018

Poniżej znajdują się wymagania dotyczące projektu. Omawiane kwestie ogólne dotyczą wszystkich trzech etapów, chyba że zaznaczono to inaczej. Następnie znajdują się szczegółowe wymagania odnośnie poszczególnych etapów. Niedostosowanie się do wymagań może spowodować obniżenie oceny.

1 Wymagania i informacje ogólne

- Projekt prowadzony jest indywidualnie. Wyjątkiem jest etap 3, w którym dopuszcza się pracę w grupach dwuosobowych za zgodą prowadzącego (dotyczy to sytuacji, gdy zakres projektu wynikający z wybranego tematu jest duży).
- Terminy oddania poszczególnych etapów oraz złożenia tematu na etap 3 znajdują się na stronie. Spóźnienie wiąże się z obniżeniem oceny. Poszczególne etapy (i temat) można złożyć wcześniej niż przewiduje to harmonogram.
- Pierwsze dwa etapy dotyczą odgórnie ustalonego tematu dla wszystkich studentów (problem ucztujących filozofów). Trzeci etap dotyczy tematu wybranego przez poszczególnych studentów (lub grupy dwuosobowe) i zatwierdzonego przez prowadzacego.
- Projekt należy pisać pod linuksem w języku `C/C++`.
- Należy korzystać z biblioteki wielowątkowej zawartej w nowszych standardach `C++` (tzn. `C++11` i wzwyż) lub wprost z biblioteki `pthreads`.
- Aplikacja powinna być pewnym rodzajem symulacji i działać bez końca do czasu wciśnięcia przycisku (np. `ESC`, `q` lub `enter`). Wyjątkiem jest etap pierwszy, w którym aplikacja może kończyć się po pewnym czasie (np. gdy każdy filozof wykona określoną liczbę cykli filozofowanie–jedzenie).
- Wątek funkcji `main` powinien zakończyć się dopiero po zamknięciu wszystkich innych wątków.

- Pierwsze dwa etapy mają niższą wagę do końcowej oceny (łącznie około 30–40%) niż etap trzeci (około 60–70%).
- Mile widziana jest możliwość uruchomienia aplikacji dla różnej liczby wątków/zasobów. Można dokonywać w tym celu rekompilacji, bardziej chodzi o dostosowanie programu do różnej liczby wątków. W przypadku braku takiej opcji należy zadbać, by wykorzystywana liczba wątków/zasobów nie była zbyt mała (dla problemu ucztujących filozofów wystarczy 5 wątków plus ewentualnie wątek graficzny, dla etapu trzeciego raczej preferowana jest znacznie większa liczba wątków).
- Kod źródłowy każdego etapu należy przesłać po uzyskaniu oceny na e-mail prowadzącego.

2 Wizualizacja

- Domyślnym sposobem wizualizacji jest wizualizacja z użyciem konsoli. Jednakże standardowy sposób pracy konsolowej (wiersz po wierszu) jest zakazany. Należy skorzystać z biblioteki `ncurses`, która pozwala na większą kontrolę konsoli (np. wypisywanie tekstu w dowolnym współrzędnych). Warunek ten nie dotyczy etapu pierwszego, który może zostać zrealizowany w standardowym trybie konsoli tj. wiersz po wierszu.
- Aby uzyskać większą liczbę barw za pomocą `ncurses` po pierwsze należy wykonać komendę `export TERM=xterm-256color` przed uruchomieniem programu. W programie należy wtedy redefiniować kolory (funkcja `init_color`), pamiętając że dostępnych miejsc na kolory jest więcej niż 8 (odpowiednia wartość jest w stałej `COLORS` i powinna wynosić 256).
- Należy pamiętać, że rozmiar okienka konsoli (domyślnie 80×24 znaki) może zostać ręcznie zwiększony, by uzyskać większą przestrzeń na wizualizację.
- Wizualizacja powinna być płynna, tj. obserwowany na ekranie (wizualizowany) stan wątku powinien zmieniać się częściej niż raz na sekundę (np. kilka razy na sekundę).
- Nie należy też przesadzać w drugą stronę – częstość aktualizacji (na ekranie) stanu wątku powinna pozwolić na śledzenie tego co się z danym wątkiem dzieje.
- W projekcie bezwzględnie zakazane jest usypianie wątków na czas bliski sekundzie lub większy – jeśli wątek musi czekać dłużej lub wykonuje dłuższą czynność (np. filozof medytuje przez dłuższy czas), to mimo wszystko należy to zamienić na więcej krótkich odcinków, by być w stanie wizualizować czas czynności (w tym przykładzie można wyświetlać dotychczasowy czas trwania medytacji, który wciąż powinien na ekranie aktualizowac się

2–4 razy na sekundę, nawet jeśli cała czynność medytacji będzie trwała kilka sekund).

- Wizualizacja powinna ukazywać stan wątków oraz zasobów. Często wykorzystywana jest jedna lub obie z poniższych technik:

  1. Przeznaczenie statycznego (cały czas w tym samym miejscu ekranu) fragmentu konsoli na wizualizację danego wątku/zasobu. Taki sposób wizualizacji dobrze nadaje się także do pokazywania postępów pracy watków (np. czas lub postęp czynności takich jak filozofowanie). Zwykle informacja taka podawana jest w formie tekstu, liczb lub pasków postępów, jednak w przypadku zasobów często spotyka się oznaczenia symboliczne (np. na każdy jednostkowy zasób przypada jeden symbol/znak).
  2. Przedstawienie stanu wątków/zasobów w postaci symboli (pojedynczych lub wieloznakowych) wraz ze zmianą ich położenia. W takich przypadkach dany obszar konsoli zwykle przedstawia widok świata (z góry lub z boku) a położenie symbolu ma ścisły związek z pozycją wątku/zasobu. Konkretny stan wątku lub postęp wykonywanej czynności można wtedy wyrazić poprzez zmianę koloru symbolu lub prostą animację (jednakże kolorem/rodzajem symbolu zwykle lepiej oznaczać rodzaj wątku).

3 Etap pierwszy

W tym etapie należy przygotować aplikację wielowątkową ukazujacą problem ucztujących filozofów ograniczając się do:

- Utworzenia wątków (filozofów).
- Symulowania "życia" filzofa (cykl filozofowanie–posiłek). Na tym etapie można założyć, że każdy filozof ma własną parę sztućców (lub wprowadzić współdzielenie sztućców bez synchronizacji zasobów, ale efektem będzie chaos/niespójność).
- Poprawnego zamykania wątków (np. po pewnym czasie).

4 Etap drugi

W tym etapie należy dokończyć aplikację ukazujacą problem ucztujących filozofów, która powinna:

- Tworzyć wątki (filozofów).
- Symulować "życie" filzofa.

- Rozwiązywać problem synchronizacji zasobów współdzielonych (sztućców) tak by nie dochodziło do niespójności.
- Rozwiązywać problem zagłodzenia.
- Rozwiązywać problem zakleszczenia (deadlock/livelock).
- Poprawnie zamykać wątki (w reakcji na wciśnięcie jakiegoś klawisza).
- W programie należy wykorzystać mutexy i, w miarę możliwości, zmienne warunkowe.

5 Etap trzeci

W tym etapie należy zrealizować te same zadania co w etapie drugim (tworzenie i zamykanie wątków, synchronizacja zasobów, zapobieganie zagłodzeniu i zakleszczeniu), jednak nie dla problemu ucztujących filozofów, lecz dla tematu wybranego przez studenta/grupę dwusobową i zatwierdzonego przez prowadzącego.

Uwagi do tematów i ich zgłaszania:

- Zgłoszenie tematu odbywa się poprzez wysłanie na e-mail prowadzącego opisu tematu w formie dokumentu PDF lub wprost w treści e-maila.
- Temat powinien zawierać nazwę (prostą, w celu identyfikacji tematu), dane studenta/grupy dwuosobowej oraz opis zasobów/wątków. Opis taki powinien określać jakie typy zasobów i wątków będą występować w projekcie i jaka będzie ich przybliżona liczba (wystarczy opis w stylu "jeden", "kilka" czy "kilkanaście"). Należy też umieścić krótki opis działania aplikacji, tzn. jaki system jest symulowany, jakie czynności wątki wykonują i jakich zasobów wymagają.
- Zabronione są tematy zbyt proste lub klasyczne (bazowy problem ucztujących filozofów, proste skrzyżowanie uliczne, bazowy problem producent-konsument), chyba że po odpowiednim rozszerzeniu/modyfikacji tematu.
- Zabronione są tematy mające na celu obliczenia równoległe (tzn. użycie wielu wątków w celu przyspieszenia jakiejś czynności np. przetwarzania obrazka).
- Praca w grupie dwuosobowej jest możliwa tylko w przypadku odpowiednio dużych projektów.
- W przypadku występowania zbyt podobnych tematów decyduje kolejność zgłoszeń.
- Aby projekt mógł być zaakceptowany wątki musza wykorzystywać zasoby współdzielone (a nie tylko prywatne) oraz muszą mieć odpowiednio bogate "życie". Można to osiągnąć poprzez jeden lub dwa *typy* wątków

odpowiednio rozbudowanych (mogących wykonywać różne czynności i korzystać z różnych zasobów) lub przez większą liczbę *typów* wątków o prostszym "życiu". Liczba wątków/zasobów jest drugorzędna.

- Wątki/zasoby różniące się tylko nazwą, ale działające/używane w ten sam sposób nie zwiększają w znaczący sposób rozbudowania projektu.