2020-11-05.

Lista programów do wykonania, 2TI, lab. SO, sem. zimowy 2020/2021

Dowolne dwa z poniższych programów stanowią element zaliczenia (2 programy po max 20 pkt. każdy). Można wykonać i przedstawić 3 programy.
Teksty specyfikacji programów pochodzą z instrukcji do laboratorium Systemów Operacyjnych.

Laboratorium 6: Podstawy pisania programów w języku C, podstawowe operacje
Ćwiczenie 5:
Napisz program, który będzie wyświetlał wartości zmiennych środowiskowych wg poniższych reguł:
- jeżeli nie podano żadnych argumentów wyświetlane są wszystkie zmienne środowiskowe
- jeżeli podano jakieś argumenty, to są one traktowane jako nazwy zmiennych, których wartości chcemy wyświetlić. Np. wywołanie ./zmienne PATH HOST LL powinno wyświetlić na ekranie wartości zmiennych PATH, HOST i LL. Jeżeli zmienna o podanej nazwie nie istnieje powinien pojawić się stosowny komunikat.

Program należy napisać za zachowaniem omówionych w materiałach reguł.

Laboratorium 7: Procesy, sygnały
Ćwiczenie 7
Należy napisać trzy programy: nowy1.c, nowy2.c oraz fkexe.c
Proces fkexe powinien po uruchomieniu utworzyć dwa procesy potomne, które następnie podmienią swoje kody na kody procesów nowy1 i nowy2. Proces macierzysty powinien poczekać na zakończenie obu procesów potomnych. W momencie zakończenia któregoś procesu potomnego, proces macierzysty wyświetla na ekranie PID tego procesu oraz kod zakończenia procesu potomnego i numer sygnału, który spowodował zakończenie potomka. Jeśli nie ma już procesów potomnych, proces macierzysty również kończy swoje działanie.
Programy nowy1.c i nowy2.c powinny wyświetlić na ekranie jakiś komunikat i zawiesić swoje działanie na, odpowiednio, 20 i 30 sekund. Dodatkowo można w tych programach dopisać obsługę sygnałów INT i TERM, która spowoduje, że w przypadku zakończenia procesu sygnałem zostanie zwrócony kod zakończenia procesu różny od 0.

Laboratorium 8: Łącza komunikacyjne i kolejki FIFO
Ćwiczenie 3:
Napisz program upipe_ex.c, który uruchomi jako osobne procesy polecenia „cat /etc/passwd” oraz „sort” oraz połączy je za pomocą łącza komunikacyjnego tak, aby uzyskać wynik identyczny z wynikami wykonania następującego wywołania z wiersza poleceń:
cat /etc/passwd | sort
Zalecane użycie funkcji dup2.

Laboratorium 9: Gniazda, komunikacja sieciowa
Ćwiczenie 4:
Napisz prosty serwer www, który na dowolne zapytanie odpowie zawartością strony www. Dla uproszczenia, nie będzie nas interesowało, jakiej strony żąda przeglądarka – zawsze wysyłamy tą samą. Dzięki temu nie musimy interpretować zapytania od przeglądarki.
Aby przeglądarka poprawnie zinterpretowała stronę należy w pierwszej kolejności wysłać ciąg znaków zgodny z protokołem http, który wygląda następująco:
HTTP/1.0 200 OK\nContent-Type: text/html\n\n
Następnie należy przesłać właściwą stronę www w postaci kolejnego ciągu znaków. Przygotowana strona testowa znajduje się w pliku site.html
Można posłużyć się programem tcps.c odpowiednio go modyfikując.

Laboratorium 10: Semafory

Ćwiczenie 4:

Skopiuj programy count1.c oraz count2.c i skompiluj. Pierwszy z nich wyświetla na ekranie liczby nieparzyste w następujący sposób:

1
3 3 3
5 5 5 5 5
itd. Po każdej wyświetlonej linii proces zasypia na czas 1 sekundy. Program count2.c wyświetla liczby parzyste w następujący sposób:
2 2
4 4 4 4
6 6 6 6 6 6
itd. Po każdej wyświetlonej liczbie proces zasypia na czas 1 sekundy. Jeśli uruchomimy te programy jednocześnie (./count1 & ./count2 &), wówczas wyniki ich wykonania będą na ekranie przemieszane.
Należy zsynchronizować te procesy za pomocą semaforów tak, aby otrzymać na ekranie następujący obraz:
1
2 2
3 3 3
4 4 4 4
itd. W obu programach oznaczono początek i koniec sekcji krytycznej. Kodu sekcji krytycznej zmieniać nie wolno. W pozostałej części programów można dopisać niezbędne operacje na semaforach. Zabronione jest stosowanie gdziekolwiek dodatkowych funkcji sleep. Wybór konwencji jest dowolny. Programy mają działać poprawnie niezależnie od tego w jakiej kolejności i jakim odstępie czasowym zostaną uruchomione. Liczbę semaforów i sposób implementacji należy dobrać tak, aby nie zachodziło niebezpieczeństwo, że jeden z procesów będzie cyklicznie wchodził do swojej sekcji krytycznej, a drugi będzie w nieskończoność czekał.

Laboratorium 11: Pamięć współdzielona

Ćwiczenie 3:

Napisz aplikację złożoną z dwóch programów: producenta i konsumenta. Procesy będą wymieniać dane poprzez segment pamięci współdzielonej. Producent zapisuje dane, a konsument odczytuje. Procesy te należy zsynchronizować za pomocą semaforów.
Struktura danych przechowywana w pamięci współdzielonej może być dowolna, powinna jednak zawierać przynajmniej jeden int i tablicę znaków. Dane do wpisania do pamięci współdzielonej producent pobiera z stdin. Po skompletowaniu danych w pamięci współdzielonej zezwala na ich odczyt przez konsumenta. Konsument wyświetla na ekranie odczytane dane, po czym zezwala producentowi na zapis nowej porcji danych, itd.

Laboratorium 12: Kolejki komunikatów

Ćwiczenie 3:

Za pomocą komunikatów zsynchronizuj działanie programów count1.c i count2.c z zajęć 10. Oczekiwany efekt i wymagania jak w zadaniu 4 z zajęć 10.
