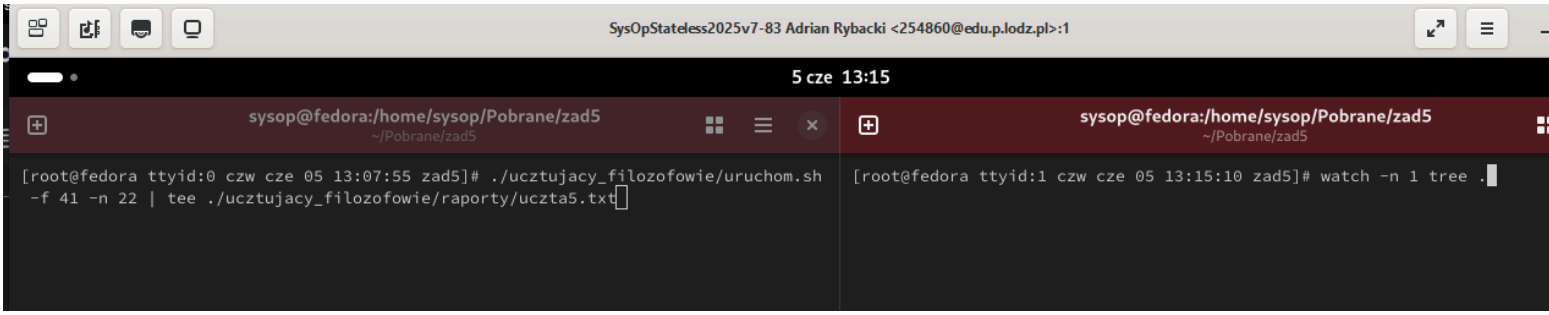
5.1.2

Początek:

Uruchomienie w powłoce bash (użytkownik nieuprzywilejowany, domyślne ustawienia, wykonanie bez żadnych błędów) każdego z trzech skryptów odpowiednio dla problemu producent-konsument, czytelnicy i pisarze oraz ucztujący filozofowie.

macie 2 okna terminali, w jednym uruchamiacie skrypt , w drugim piszecie watch -n 1 tree sciezka_do_katalogu
watch -n 1 <------ co ile sekund ma się odświeżać



1. **Rodzaje przetwarzania współbieżnego, ze względu na liczbę wykorzystywanych jednostek przetwarzających.**

Teoria:

Przetwarzanie współbieżne zbioru procesów/wątków sekwencyjnych jest ciągiem następujących kolejno stanów i zdarzeń, występujących w ramach wykonania rozkazów z różnych zadań (jeden proces/wątek może się zacząć przed zakończeniem poprzedniego).
- przetwarzanie wątku może zacząć się przed zakończeniem poprzedniego (kiedy je nazywamy wpsolbieznymi)

Przetwarzanie równoległe - gdy ich zadania są wykonywane jednocześnie z wykorzystaniem różnych jednostek przetwarzających (przy czym nie jest możliwe równoległe przetwarzanie włókien).

Czyli **podsumowując** jeśli mamy n filozofów, to żeby osiągnąć przetwarzanie równoległe, potrzebujemy co najmniej n + 1 jednostek CPU, ponieważ istnieje 5 procesów (filozofów) i proces macierzysty, który tworzy te procesy filozofów.

Przetwarzanie pseudorównoległe (tzw. **przeplot**) – jest to przykład zwielokrotnienia w czasie jednostki zasobu CPU, a występowanie rywalizacji zadań gotowych do wykonania o przydział jednostki przetwarzającej zapewnia pseudorównoległość. (bardzo szybko przelaczaja sie miedzy jednostkami

przetwarzajacymi, co sprawia wrazenie dla uzytkownika ciaglosci). Podział w domenie czasu na kwanty czasu.

2. Kryteria poprawności programu współbieżnego.

Teoria:

Program współbieżny - program w wyniku uruchomienia którego powstają co najmniej 2 wątki współbieżne.

Bezpieczeństwo (safety) – w trakcie przetwarzania zadań zawsze wystąpią jedynie zdarzenia dopuszczalne. Dopuszczalne nie narusza regul przetwarzania danych. Czyli nie wystąpią zdarzenia niedopuszczalne.
Zadanie **niedopuszczalne** to zdarzenie, które nie spełnia reguł przetwarzania danych *(np. 2 filozofów podnosi ten sam widelec)*.

Zdarzenie dopuszczalne - zdarzenie, ktore moze pojawic sie w okreslonym stanie przetwarzania sekwencyjnego watku procesu i nie narusza ustalonych regul przetwarzania danych (zgodnie z ustalona **specyfikacja funkcjonalna - zbior regul przetwarzania danych** oprogramowania).

Żywotność (liveness) - po skończonej liczbie zdarzeń przetwarzanie kolejnych instrukcji programu zostanie kontynuowane. Czas oczekiwania nie jest nieskonczony, gwarantuje, ze osiagniemy jakis postep w skonczonym czasie. Żywotność oznacza, że jeśli pewne zadanie spełnia warunki do rozpoczęcia wykonania, to w odpowiednim czasie zostanie ono wykonane. Dla **każdego** ze współbieżnie przetwarzanych zadań musimy obserwować **postęp** w realizacji zadania w **skończonym** czasie.

3. Który z warunków poprawności narusza zagłodzenie i uwięzienie oraz jaka jest różnica pomiędzy zagłodzeniem i uwięzieniem?

Teoria:

Zagłodzenie (starvation) – przetwarzane współbieżnie zadanie nieprzerwanie przebywa w stanie zablokowanym. Takie zadanie nie ma dostępu do zasobu (np.

jednostki przetwarzającej) przez nieskończony okres czasu. Proces **nie ma dostępu** do jednostki przetwarzającej lub do innego współdzielonego zasobu. (proces ma wszystko gotowe, ale **nigdy nie dostaje zasobu**)

Uwięzienie (livelock) – przetwarzane współbieżnie zadanie uzyskuje przydział jednostki przetwarzającej lecz bez postępu w realizacji zadania. Takie zadanie ma dostęp do zasobów, ale działa w nieskończoność i nie czyni żadnego postępu. Proces **ma dostęp** do jednostki przetwarzającej,ale nie robi żadnego postępu w przetwarzaniu zadania.

Zagłodzenie i uwięzienie naruszają, który z tych warunków bezpieczeństwa czy żywotności?
Oba żywotności.

Mechanizm synchronizacji polega na tym, że pewne procesy czekają na postęp innych procesów, żeby nie doszło do zagłodzenia

To jak to jest ma przydział jednostki i nie ma realizacji zadania?
Może tak być np. nieskończona pętla while.

Co musi wystąpić dla zadania żeby to zadanie mogło się ukończyć?
Postęp realizacji zadania.

4. **Wyjaśnienie pojęć: hazard/wyścig, przeplot, konflikt. Wskazanie unikalnego wyścigu dla zadań ucztujących filozofów.**

Teoria:

Hazard / wyścig - sytuacja, w której operacje między przetwarzanymi zadaniami powodują, że w **zależności od kolejności występowania zdarzeń system operacyjny uzyska różne stany** (np. dwoch filozofow sciga sie o ten sam widelec)

przyklad zdarzenia zewnetrznego - procesowi nie jest przydzielona jednostka przetwarzająca (przeplot)

Który element odpowiada za przydzielanie jednostki przetwarzającej? - planista krótkoterminowy

Przeplot - globalny porządek wykonywania rozkazów różnych zadań (podział czasu jednej jednostki przetwarzającej) (inaczej przetwarzanie pseudorównoległe. patrz pyt. 1) (jest mniej jednostek niz procesow i wymieniaja sie jednostka przetwarzajaca)

Konflikt - sytuacja, kiedy co najmniej dwa zadania przetwarzane współbieżnie będą operować na wspólnym zasobie i przynajmniej jedno z nich dokona operacji zapisu (modyfikacji tego zasobu)

Przykłady wyścigów:

1. Jeden filozof próbuje podnieść widelec, gdy drugi go odkłada
2. Dwóch filozofów sięga po ten sam widelec
3. Dotarcie do bariery, w zależności który dotrze pierwszy, ten będzie czekał na innych i będzie właścicielem blokady
4. Dwóch filozofów jednocześnie próbuje założyć blokadę na pliku bariery
5. Jeden filozof zaczyna jeść, gdy drugi jeszcze nie odłożył widelca
6. Jeden filozof już kończy jedzenie i zwalnia widelec, podczas gdy drugi dopiero po niego sięga
7. Filozof A wchodzi do bariery i sprawdza stan potoku, zanim filozof B zdąży się wpisać
8. Jeden filozof losuje czas rozmyślania krótszy niż drugi i szybciej próbuje sięgnąć po widelec
9. Dwóch filozofów kończy rozmyślanie w tym samym czasie i próbuje podnieść ten sam widelec
10. Filozof, który szybciej zje połowę posiłków, jako pierwszy przejmuje kontrolę nad barierą
11. Filozof zapisuje się do potoku, gdy inny właśnie go odczytuje
12. Dwóch filozofów jednocześnie chce podnieść lewy widelec, który należy do ich sąsiada
13. Filozof kończy jeść i zwalnia widelec dokładnie wtedy, gdy drugi zaczyna go podnosić

5. Gdzie zapisane jest zadanie filozofa i jaki to jest rodzaj zadania?

Teoria:

Zadanie filozofa zapisane jest w funkcji filozof() i jest to zadanie uruchomione w tle, jako osobny proces.

Skoro parametry funkcji wpływają nam na zadanie to mamy do czynienia z **zadaniem sparametryzowanym.**

Funkcja filozof() reprezentuje zadanie do którego w wywołaniu funkcji zostają przekazane parametry takie jak np. które widelce znajdują się obok filozofa. Zadanie uruchamia się jako osobne procesy jednowątkowe (dekompozycja problemu na zadania implementowane jako osobne procesy) , ponieważ na

końcu linijki z wywołaniem funkcji filozof znajduje się znak “&” co oznacza, że zadanie(funkcja filozof) zostanie uruchomione w tle i jako osobny proces.

Gdzie mamy tutaj dekompozycje na zadania przykład zadania wydzielonego w tym skrypcie?
Funkcja filozof()

Jak to jest możliwe, że z jednej funkcji jesteśmy w stanie utworzyć wiele zadań?
Ponieważ funkcja jest sparametryzowana

Czy każdy parametr tej funkcji jest parametrem zadania?
Nie, tylko numer filozofa, plik pierwszego widelca oraz plik drugiego widelca

6. **W jaki sposób powoływane są procesy realizujące zadania poszczególnych filozofów i jaką mają charakterystykę?**

Teoria:

Są one powoływane jako **procesy w tle**. Świadczy o tym znak **'&'.**

Każdy z powołanych procesów dostanie swoje zadanie filozofa do wykonania,dzięki parametrom te zadania będą rozróżnione.

7. **Ile będzie utworzonych procesów dla ustalonej przez nauczyciela liczby filozofów i który z procesów będzie kończył się jako ostatni? - wymagane wskazanie odpowiedniej instrukcji w skrypcie ucztujących filozofów.**

Teoria:

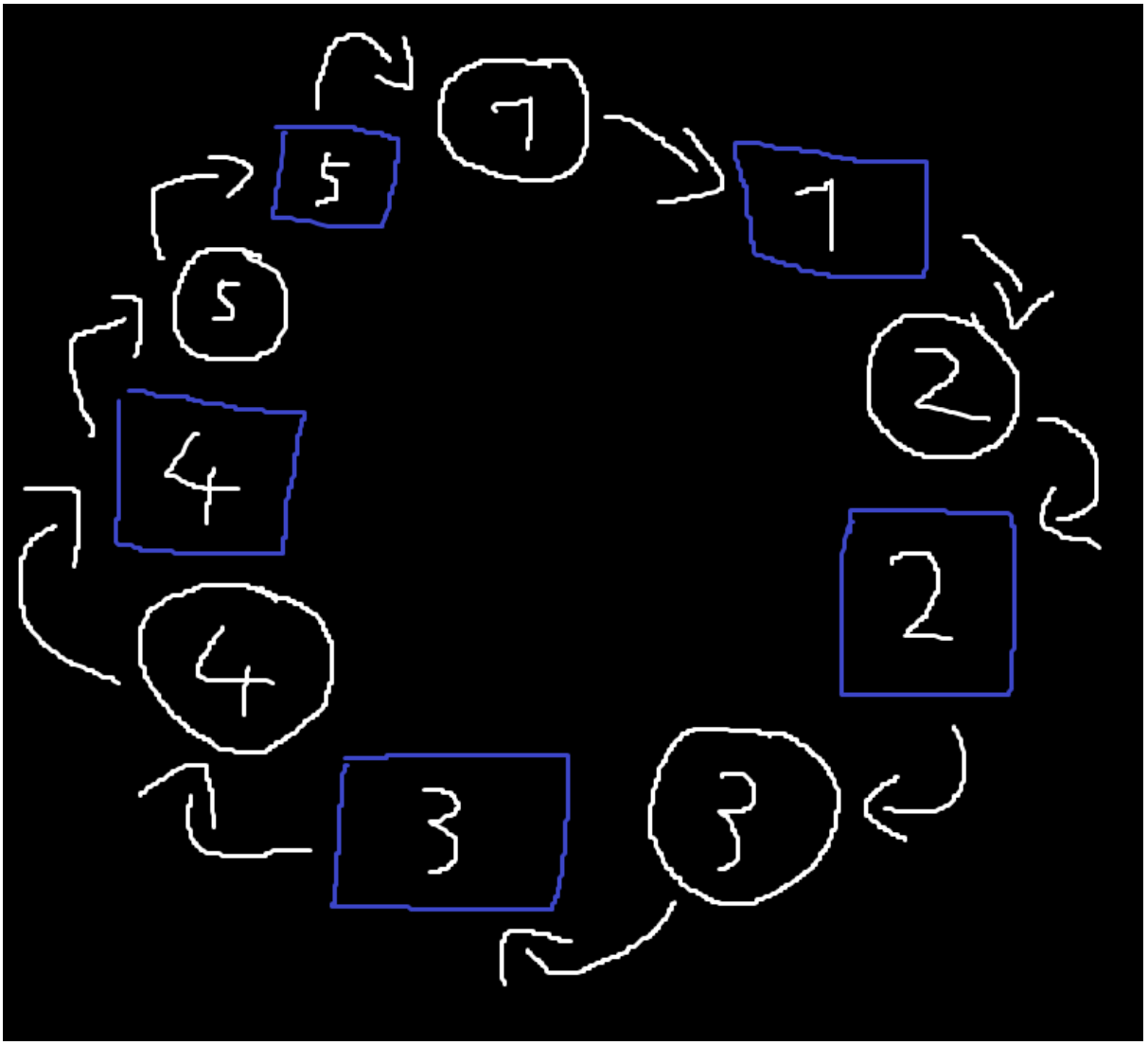
Dla podanej liczby filozofów utworzonych będzie tyle procesów, **ile mamy filozofów + 1**, ponieważ powstaje jeszcze jeden proces, który wywołuje procesy filozofów. Wszystkie te procesy są jednowątkowe. Jako ostatni zakończy się proces główny. Pokazuje to instrukcja „**wait**”

wait - proces wstrzymuje swoje działanie do momentu, kiedy wszystkie procesy potomne utworzone przez proces macierzysty się zakończą.

usunięcie wait spowoduje - Przekładając na ucztę nie wszyscy filozofowie zdążą zjeść

5.1.3

UWAGA, UWAGA - GRAFA TRZEBA ZROBIĆ



1. **Jak zauważyć problem zakleszczenia w przygotowanym grafie?**

Teoria:

Zakleszczenie - sytuacja wzajemnego zablokowania się procesów poprzez alokację części zasobów potrzebnych innym procesom. I żaden z nich nie zwalnia swoich zasobów (bo nie ma wywłaszczania z zewnątrz)

Tutaj będzie potrzebny graf do przedstawienia problemu zakleszczenia, można go zauważyć w taki sposób, że każdy proces oczekuje na jakiś zasób (widelec), który jest przydzielony do innego procesu. Dzieje się to w cykliczny sposób, który tworzy nieskończona pętle oczekiwania na zasob. Zakleszczenie jest wtedy, kiedy procesy nie wykonują swoich zadań, bo nie mają przydzielonych zadań/procesów, ale nigdy się nie doczekają.

Modyfikacja skryptu aby nastąpiło zakleszczenie:

URUCHOM.SH **<- ten w filozofach**

SPRAWDŹCIE CZY MODYFIKUJECIE I URUCHAMIACIE DOBRY SKRYPT (DOBRA ŚCIEŻKA JEST)

PRZED

```
for NUMER_FILOZOFA in $(shuf -i1-$LICZBA_FILOZOFOW)
do
    PIERWSZY_WIDELEC=$NUMER_FILOZOFA
    DRUGI_WIDELEC=$(((${NUMER_FILOZOFA}+1)%(${LICZBA_FILOZOFOW}+1)))

    if [[ $DRUGI_WIDELEC -eq 0 ]]
    then
        DRUGI_WIDELEC=$PIERWSZY_WIDELEC
        PIERWSZY_WIDELEC=1
    fi
```

PO

```
for NUMER_FILOZOFA in $(shuf -i1-$LICZBA_FILOZOFOW)
do
    PIERWSZY_WIDELEC=$NUMER_FILOZOFA
    DRUGI_WIDELEC=$(((${NUMER_FILOZOFA}+1)%(${LICZBA_FILOZOFOW}+1)))

    if [[ $DRUGI_WIDELEC -eq 0 ]]
    then
        DRUGI_WIDELEC=1
    fi
```

2. **Ilu zadań dotyczy zakleszczenie i jak jego wystąpienie wpływa na poprawność programu?**

Teoria:

Zakleszczenie w tym przypadku dotyczy pięciu zadań (i pięciu zasobów). Występuje zakleszczenie, stan nieprzerwanego wzajemnego blokowania się zadań, które nie wykazują postępu. System oczekuje na zajście zdarzenia, które nigdy nie zajdzie. Występuje ciągłość blokowania wynikająca z wykorzystywania zasobów o dostępie wyłącznym, które nie zostają zwolnione w wyniku wstrzymania wykonania zadania. Łamana jest zasada żywotności (zadania muszą wykazać postęp w określonym czasie).

Aby występowało zakleszczenie muszą być przynajmniej dwa zadania.

3. **Warunki konieczne wystąpienia zakleszczenia i który z nich został wyeliminowany poprzez zastosowane rozwiązanie eliminujące zakleszczenia filozofów?**

Teoria:

Warunki konieczne:

- **Wzajemne wykluczanie**
  zasób może być w danej chwili wykorzystywany tylko przez jedno zadanie i nie może być dostępny dla innych zadań (dostęp wyłączny).

- **Brak wywłaszczeń**
  zwolnienie zasobu może być zrealizowane tylko z inicjatywy zadania, które wykorzystuje ten zasób (wywłaszczenie - wymuszone przełączenie kontekstu)

- **Przetrzymywanie i oczekiwanie**
  zadanie posiadające przydzielony zasób może żądać jednostek innego zasobu, które są przydzielone innemu zadaniu.

- **Cykl oczekiwania**
  istnienie zbioru zadań, w którym można wyróżnić cykl, w którym kolejne pary zadań są powiązane poprzez zależność oczekiwania i przetrzymywania zasobów (eliminujemy to tym, ze biora najnizszy widelec) bo jakby kazdy wzial swoj numer to by nie bylo problemu.
  relacja między kooprocesami (zamknięty obieg)

Hierarchia zasobow definiuje kolejnosc alokacji zasobow (w naszym przypadku widelcow).

*Są to **warunki konieczne**, ale **nie muszą** one oznaczać wystąpienia zjawiska zakleszczenia, a brak spełnienia któregokolwiek z tych warunków oznacza brak zakleszczenia.*
Ten warunek jest konieczny, ale niewystarczający do wystąpienia zakleszczenia

4. Jak zastosować hierarchię zasobów i dwufazowe blokowanie.

Teoria:

Hierarchia zasobów - wprowadzenie globalnego porządku dla zasobów wspólnych o dostępie wyłącznym np. poprzez numerowanie i ustalenie reguły dostępu do tych zasobów. Np. jeżeli zadanie ma przydzielony n-ty zasób to nie może być mu przydzielony zasób mniejszy od n.

W tym problemie to określenie kolejności alokowania zasobów

Np. proces nie może otrzymać jednostek zasobu, jeśli przydzielono mu jednostki zasobu występujące wcześniej względem ustalonego porządku. Jednostki tego samego zasobu muszą być zamawiane w jednym zleceniu.

Na przykładzie ucztujących filozofów, nie dojdzie do zakleszczenia, ponieważ hierarchia zasobów wpływa na to tak: filozofowi 5 zostanie przydzielony widelec 1 a potem widelec 5 przez co filozof 1 nie będzie mógł wziąć widelca 1.

(filozof 5 po pobraniu widelca 5 nie może pobrać widelca 1, ale może pobrać najpierw widelec 1 a potem widelec 5).
Hierarchia eliminuje cykl, ponieważ każdy proces może żądać zasobów tylko w jednym kierunku.

Dwufazowe blokowanie - protokół, w ramach którego wyróżnia się fazę blokowania niezbędnych zasobów oraz fazę przetwarzania z wykorzystaniem tych zasobów i zwracanie ich do systemu operacyjnego. Zadanie najpierw blokuje wszystkie wspólne zasoby zwrotne i dopiero wtedy następuje przejście do fazy drugiej. Wszystkie jednostki zasobu muszą być zamawiane w jednym zleceniu.

lib uczta filozofow

- Pierwsza faza: podnoszenie widelców

```
podnies_widelec $1 $6 $DESKRYPTOR_WIDELEC_1
podnies_widelec $1 $7 $DESKRYPTOR_WIDELEC_2
```

- Druga faza: inkrementacja liczby zjedzonych posiłków, jedzenie i odkładanie widelców

```
odloz_widelec $1 $7 $DESKRYPTOR_WIDELEC_2
odloz_widelec $1 $6 $DESKRYPTOR_WIDELEC_1
```

Faza 1 – blokowanie: proces próbuje zdobyć **wszystkie zasoby**, które są mu potrzebne.
Faza 2 – wykonywanie: dopiero gdy ma **wszystkie zasoby**, wykonuje działanie, a potem **oddaje zasoby**.

Zwrot zasobów ma znaczenie - zwalniane w odwrotnej kolejności niż podnoszenie - tak się dzieje.
Wtedy **żaden proces nie może trzymać zasobu wyżej w hierarchii i próbować zdobyć niższego** — a to właśnie **kluczowy warunek zakleszczenia (cykl oczekiwania)**.

Zasoby o dostępie wyłącznym są przykładami zasobów z grupy zasobów wspólnych.

Aby wyeliminować zakleszczenie za pomocą hierarchii zasobów należy ponumerować zasoby o dostępie wyłącznym i ustalić regułę np. że nie można otrzymać zasobu niżej w hierarchii, jeśli wcześniej otrzymano przydział zasobu wyżej w hierarchii.

5. **Czym jest współbieżność konkurencyjna i współbieżność kooperacyjna? - wymagane wskazanie, gdzie występuje każda z tych współbieżności pomiędzy zadaniami filozofów?**

Teoria:

Konkurencyjna – wątki i procesy nie współpracują ze sobą, a wzajemne oddziaływanie pomiędzy nimi sprowadza się do rywalizacji o dostęp do zasobów, których potrzebują do realizacji przetwarzania swoich zadań.

Przykładem jest podnoszenie widelców, dwóch filozofów konkuruje o widelec.

Dwaj filozofowie chcą sięgnąć jeden widelec i to są te same sytuacje. Która instrukcja podnosi 5 widelec przez 4 filozofa a która 5 widelec przez filozofa o numerze 5? Druga instrukcja podnieś widelec dla obu.

```
podnies_widelec $1 $6 $DESKRYPTOR_WIDELEC_1
podnies_widelec $1 $7 $DESKRYPTOR_WIDELEC_2
```

Kooperacyjna – wątki i procesy, zadania współpracują ze sobą, poprzez komunikację i synchronizację koordynują wykonywanie swoich zadań. Współbieżność kooperacyjna, komunikacja przy użyciu potoku nieanonimowego – echo >$5 wysłany pusty wiersz, cat$5 – pusty komunikat.

```
function zaczekaj_na_innych() { #numer_filozofa liczba_filozofow deskrypto
flock -x -n $3
    if [[ $? -eq 0 ]]
    then
        komunikat $1 "Założyłem blokadę wyłączną na pliku $4"
    LICZNIK=$2
    while [[ $LICZNIK -gt 1 ]]
    do
        LICZNIK=$(($LICZNIK-$(cat $5|wc -l)))
    done
    else
    komunikat $1 "Nie udało się założyć blokady wyłącznej na pliku blokady
    echo >$5
    komunikat $1 "Zakładam blokadę wyłączną na pliku $4"
    flock -x $3
        komunikat $1 "Założyłem blokadę wyłączną na pliku $4"
    fi
    flock -u $3
    komunikat $1 "Zdjąłem blokadę wyłączną z pliku $4"

}
```

Mechanizm synchronizacji polega na tym, że pewne procesy czekają na postęp innych procesów, żeby nie doszło do zagłodzenia

memoliada


FAJNE TE SYSOPY
TAKIE NIE ZDANE

PYTANIA

Co ma na celu to zadanie?

O co dokładnie chodzi z unikalnym wyścigiem? Punkt 4

Charakterystyka zadań? Punkt 6

Skryba:

Trza uruchomić te skrypty

numer indeksu modulo 7

na drugim terminalu pokazać jak tworzą się pliki w strukurze (tree, watch)
ważne dla konsumenta i producenta
w przypdaku filozofów widelce

odpowiadamy na pytania nie z listingów ale coś tam linijki ze skryptów

wykazywanie z filozofa poszczególne linijki jako taki “listing” to w dalszej części

jak działa wait - czeka na śmierć wszystkich procesów potomnych

jak są te unikalne coś tam krasiuk nie wie
nie pokazywać że uniklane tylko pokazać przykład wyścigu
mechanizm synchronizacji

rozwiązanie probblemy tam mamy a potrzebujemy sam problem (coś tam wpis z forum)

charakterystyka procesów tego też nie wie ojoj
proces zparametryzowany, sięgnąć do notatek wykładowych

Graf można przygotować przed zajęciami (podobnie jak na wykładzie)
ponumerować procesy i widelce i zakleszczenie scharakteryzować
Wykazać że wiesz co to cykl i przerwanie tego
🤓☝️
Wymagane jest rozumienie wszystkich skryptów

funkcje filozofa jak sie uruchamia te sprawy

C===3