# 1. Klokun

Номер в списку групи $S = 8$, кількість літер у прізвищі $L = 6$.

1. $1 + (S \mod 12) = 1 + (8 \mod 12) = 1 + 8 = 9.$
2. $1 + ((S + L) \mod 11) = 1 + ((8 + 6) \mod 11) = 1 + 3 = 4.$
3. $1 + ((S + 2L) \mod 18) = 1 + ((8 + 12) \mod 18) = 1 + 2 = 3.$

## 1.1. Чи є такий стан системи безпечним або небезпечним.

Нехай в деякій комп'ютерній системі виконуються 4 процеси, які спільно використовують 20 ресурсів. Розгляньте наступну таблицю розподілу ресурсів між процесами:

| Номер процесу, I | Максимальна потреба, МАКС[I] | Виділено, ВИД[I] | Залишок, ЗАЛ[I] |
|---|---|---|---|
| 1 | 7 | 6 | 1 |
| 2 | 8 | 3 | 5 |
| 3 | 5 | 4 | 1 |
| 4 | 7 | 3 | 4 |

Чи є такий стан системи безпечним або небезпечним? Доведіть свою відповідь.

Безпечним називається стан, в якому хоча б одна послідовність подій не викличе взаємоблокування.

Щоб визначити, чи є стан безпечним, спочатку визначимо, скільки всього ресурсів виділено. Для цього сумуємо виділені ресурси усіх процесів:

$$\text{ВИД} = 6 + 3 + 4 + 3 = 16.$$

Тепер визначимо, скільки ресурсів є у резерві. Для цього знайдемо різницю загальної кількості ресурсів (ВСЬОГО $= 20$) і кількості виділених:

$$\text{ВСЬОГО} - \text{ВИД} = 20 - 16 = 4.$$

Отже, в резерві 4 ресурси.

Щоб виконати процеси 1 і 3, досить 1 ресурса (залишок ЗАЛ$[I]$). Кожен процес після свого

виконання звільнить усі виділені ресурси. Отже, якщо з резерву (4 ресурси) виділити 1 ресурс процесу 1, він виконається, звільнить 7 ресурсів, і ресурсів стане:

$$4 - 1 + 7 = 10.$$

Цих 10 вільних ресурсів у резерві достатньо, щоб виконати будь-який залишившийся процес, тому даний стан не приведе до взаємоблокування і є безпечним.

## 1.2. Яка функція довготермінового планувальника робіт (job scheduler)?

### 1.2.1. Варіант 1

Функція довготермінового планувальника робіт у тому, щоб приймати рішення, чи додавати роботу (процес) у пул процесів, що виконуються в системі, тобто чи запускати роботу на виконання. (https://www.intuit.ru/studies/courses/631/487/lecture/11055?page=4)

Рішення може бути прийняте на основі таких умов, як настання певного часу, як це робить Cron в Unix і `taskschd.msc` у Windows.

### 1.2.2. Варіант 2

Функція довготермінового планувальника робіт у тому, щоб запускати певну робіту, коли виконується умова її запуску. Такими умовами можуть бути настання певного часу або настання певної події. (Wiki, не виключає перше визначення).

Прикладом довготермінового планувальника робіт є Cron в операційних системах GNU/Linux і «Планувальник робіт» — у Windows. Обидва планувальники можуть запускати робіті у певний час, певні дні або із заданою періодичністю.

## 1.3. Охарактеризуйте явище процесу з погляду ядра операційної системи. Які події відбуваються під час створення процесу?

### 1.3.1. Варіант 1

З точки зору ядра операційної системи, процес — це екземпляр програми, представлений у вигляді спеціальної структури даних, яка містить інформацію про даний процес:

- стан процесу, лічильник інструкцій, зміст регістрів процесора;

- дані, необхідні для планування використання процесору та управління пам'яттю (пріорітет процесу, розмір і місцезнаходження адресного простору);
- облікові дані (ідентифікатор процесу, користувач, який його запустив і так далі);
- відомості про пристрої вводу-виводу (список закріплених за процесом пристроїв, таблиця відкритих файлів).

Назвемо таку структуру блоком керування процесом (БКП).

Під час створення процесу відбуваються такі події:

1. Створюючи процес, система створює новий БКП зі станом «створюється» і надає йому ідентифікаційний номер.
2. Система виділяє ресурси процесу.
3. Система заносить програмний код в адресний простір процесу, встановлює значення даних і лічильника інструкцій.
4. Система заповнює БКП залишившоюся службовою інформацією, що залишилась, і встановлює стан процесу «готовий до виконання».

### 1.3.2. Варіант 2

З точки зору ядра операційної системи, процес — це екземпляр програми, що виконується. Ці екземпляри представлені у вигляді структур даних, що містять інформацію про процес: значення регістрів, лічильника інструкцій, слова стану програму, вказівника на стек та інше.

Усі процеси створюються за рахунок існуючих, тобто батьківський процес створює дочірній. Створення дочірнього процесу на прикладі системного виклику `fork()` відбувається так:

1. Виконується перевірка, чи може батьківський процес створити дочірній. Якщо так, переходимо далі.
2. Стан і ресурси ядра батьківського процесу копіюються у дочірній, встановлюючи ідентифікатори створеного дочірнього процесу і батьківського процесу, який його створив. Статистичні дані про процес обнуляються.
3. Копіюється стан виконання батьківського процесору: зміст регістрів, лічильника інструкцій, стек і так далі.
4. Встановлюються початкові значення регістрів.

5. Дочірній процес «прокидається» і стає доступним для виконання.

Після створення нового процесу, він має власний екземпляр адресного простору, який є точною копією батьківського.

## 1.4. Література

### 1.4.1. Безпечний стан

- http://novostynauki.com/e-ntsiklopediya/lektsii/sistemnoe-programmnoe-obespechenie/obhod-tupikov-algoritm-bankira-obnaruzhenie-tupikov-i-vosstanovlenie-sistemy/

### 1.4.2. Процес

- https://www.intuit.ru/studies/courses/631/487/lecture/11055?page=3
- https://www.quora.com/Linux-Kernel/In-Linux-after-a-fork-where-exactly-does-the-childs-execution-start-If-the-childs-program-counter-is-the-same-as-the-parents-does-it-start-executing-the-same-kernel-code-that-the-parent-was-executing-or-does-it-start-executing-in-user-space
- http://www.informit.com/articles/article.aspx?p=370047&seqNum=2