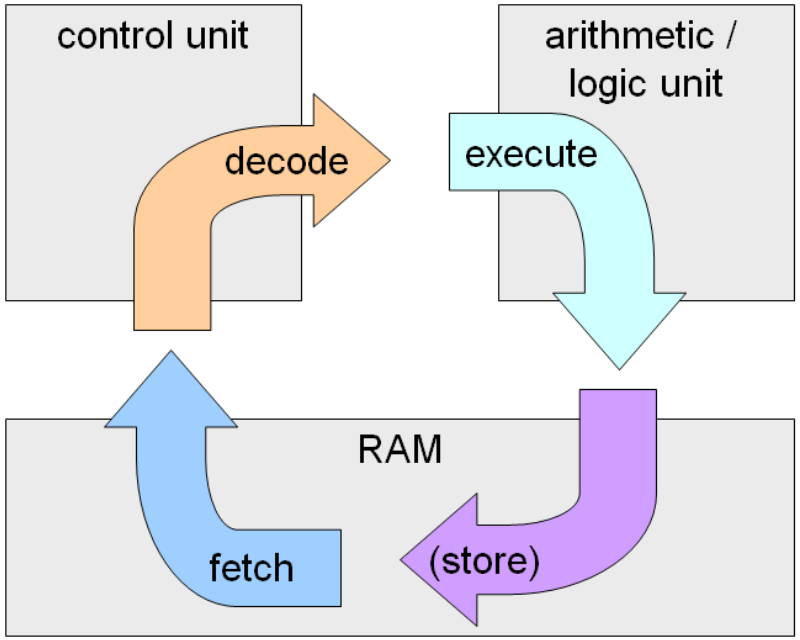
Лекція 3. Управління обчислювальними процесами (1)

01. Організація обчислювальних процесів
02. Блок управління процесом (Process Control Block)
03. Паралельне виконання обчислювальних процесів
04. Переключення контексту процесів (Context Switch)
05. Запуск та зупинка процесів
06. Рекомендована література

01.1. Організація обчислювальних процесів



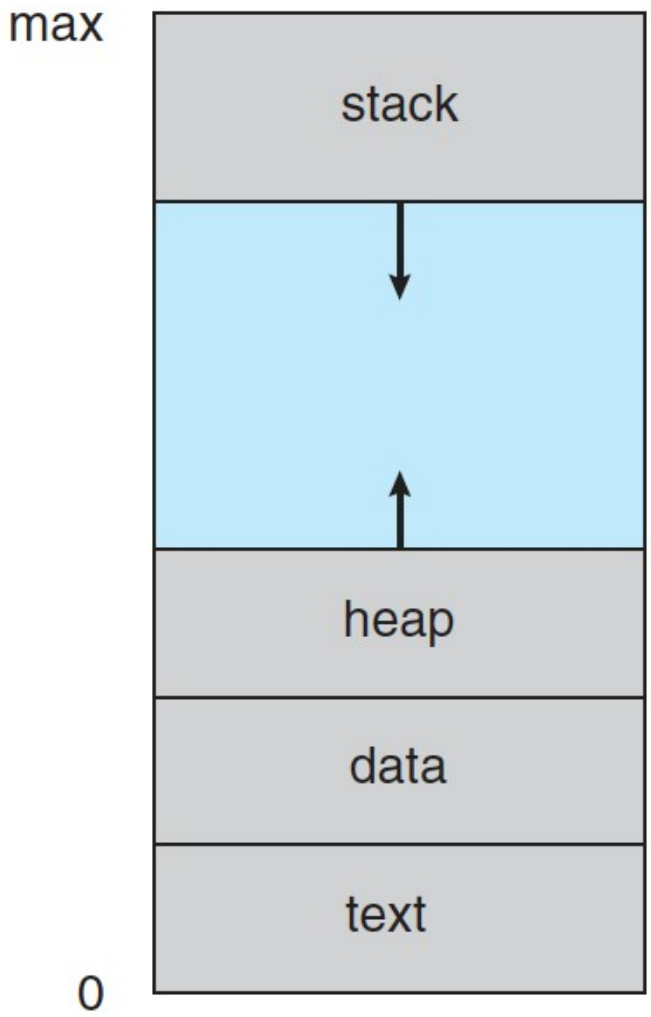
1. Обчислювальний процес (process) – це примірник деякої програми під час її виконання.

2. Окремий процес складається з послідовності машинних інструкцій, які виконуються CPU.

3. Варіанти:

1) одна програма = один процес;
2) одна програма = багато процесів.

01.2. Організація обчислювальних процесів



Організація пам’яті процесу:

1) сегмент коду (text),

2) сегмент даних (data),

3) сегмент стеку (stack),

4) heap (пам’ять, яка виділяється процесу в режимі dynamic memory allocation).

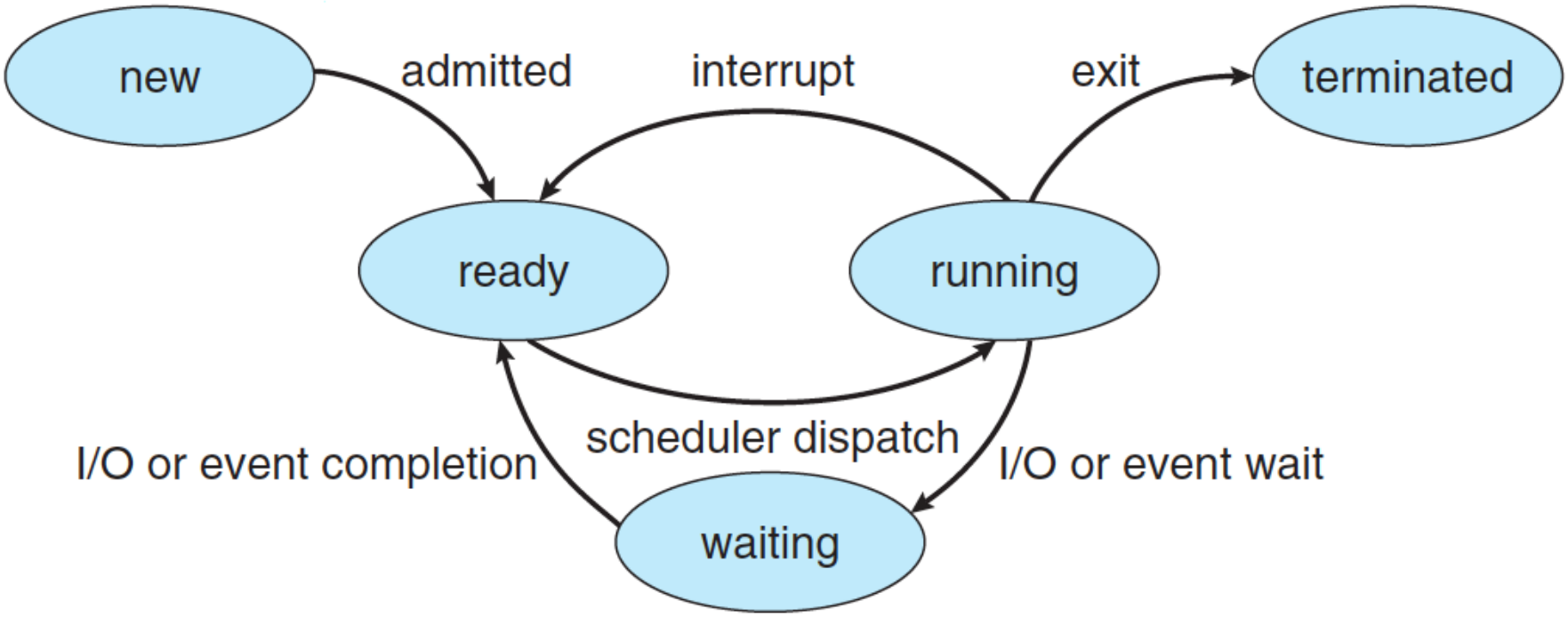
01.3. Організація обчислювальних процесів

Багатозадачність (multitasking):

1) розділення в часі (time-sharing system):
один обчислювач = багато процесів → «штучний» паралелізм;

2) розділення у «просторі» (parallel system):
багато обчислювачів = багато процесів;
+ спільне використання процесами декількох комп'ютерів, процесорів, процесорних ядер → «справжній» паралелізм

01.4. Організація обчислювальних процесів



Граф станів процеса (Diagram of process states)

01.5. Організація обчислювальних процесів

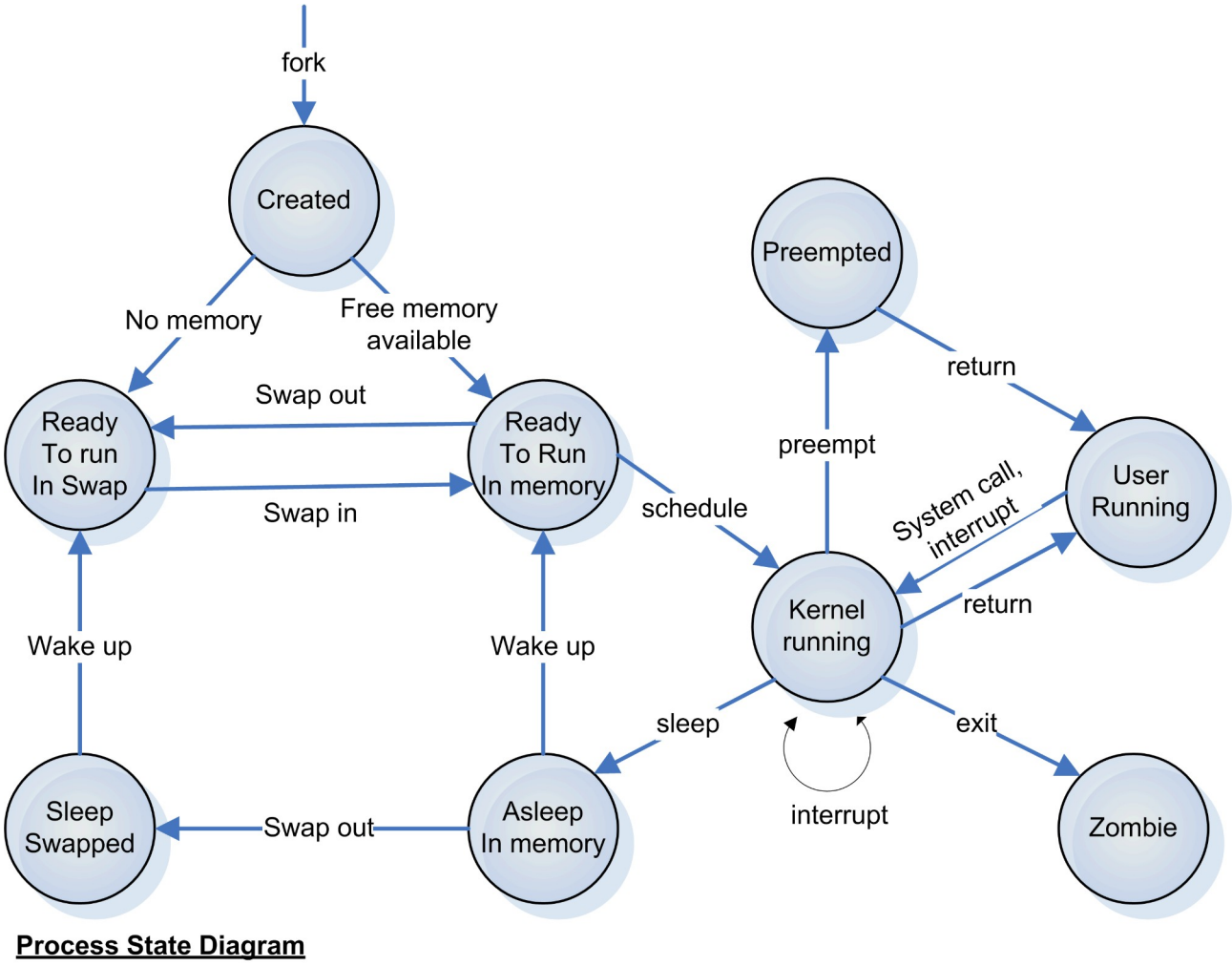
Стани процесу:

1) виконання (процесор виконує послідовність машинних команд даного процесу);

2) очікування (процес не виконується, оскільки заблокований «своїми» «внутрішніми» причинами, тобто очікує на подію, наприклад, на збільшення значення семафора);

3) готовність до виконання (процес не виконується у зв’язку з «зовнішньою» причиною: процесор зайнятий іншим процесом).

01.6. Організація обчислювальних процесів



Граф станів процесу в ОС UNIX

01.7. Організація обчислювальних процесів: Основні завдання

1. Облік процесів та зв'язків між ними (Process Control Block, task list, дерево процесів).
2. Забезпечення паралельного виконання процесів (організація, переключення контексту, алгоритми планування, реалізація алгоритмів планування).
3. Запуск та завершення процесів.
4. Синхронізація паралельних процесів.
5. Забезпечення обміну даними між процесами (IPC: inter-process communication).
6. Підтримка програмних потоків (threads).
7. Забезпечення виконання процесів в розподілених системах (distributed systems).

02.1. Блок управління процесом (Process Control Block)

Блок управління процесом (Process Control Block (PCB)) - це структура даних ядра ОС, яка містить всю необхідну інформацію про окремий процес,

в тому числі:

1) стан процеса (новий, готовий до виконання, виконується, очікування, завершений);

2) вміст лічильника команд процесора (адреса машинної інструкції, яка буде виконуватись наступною);

02.2. Блок управління процесом (Process Control Block)

3) вміст регістрів процесора (різні варіанти в залежності від моделі процесора; в загальному: вміст службових регістрів та регістрів загального призначення);

4) інформація про диспетчеризацію процеса (значення пріоритету, посилання на черги диспетчера та інші параметри);

5) інформація про управління пам’яттю процеса (значення бази для розрахунку фізичних адрес, таблиця сторінок, таблиця сегментів та ін.);

02.3. Блок управління процесом (Process Control Block)

6) облікова інформація (загальний час виконання, час використання процесора, кількісні показники використання ресурсів та ін.)

7) інформація про ввід/вивід (список пристроїв вводу/виводу, з якими працює процес; список відкритих процесом файлів та ін.).

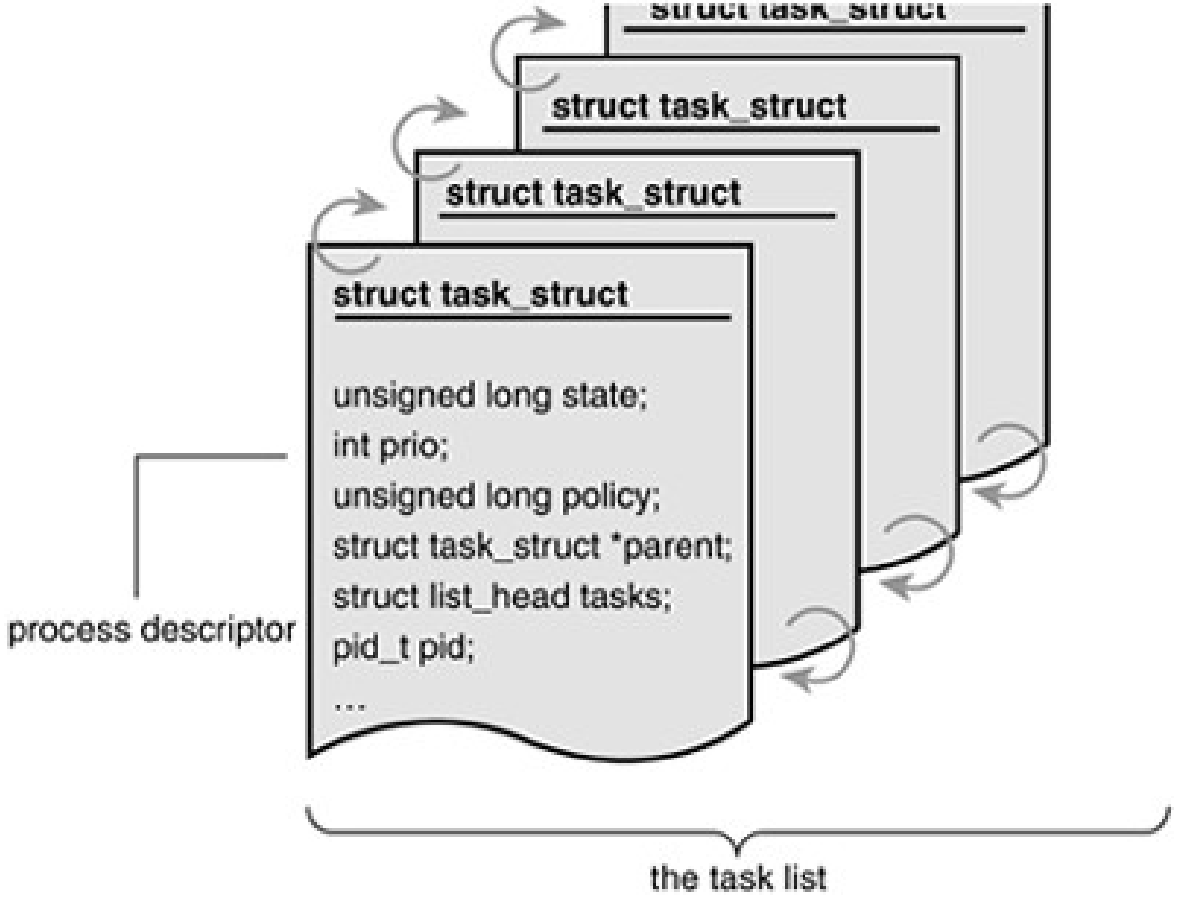
02.4. Блок управління процесом (Process Control Block): ОС Linux

1. Блок управління процесом = process descriptor (дескриптор процесу) = примірник структури: struct task_struct (оголошена у <linux/sched.h>, один примірник займає в пам'яті ~1.7kb);

2. Дескриптори процесів об'єднані в кільцевий двонаправлений (circular doubly linked) список процесів (task list);

3. На множині всіх процесів визначено бінарне відношення «батько-син» (parent-child), на основі якого будується дерево процесів (tree of processes).

02.5. Блок управління процесом (Process Control Block): ОС Linux



02.6. Блок управління процесом (Process Control Block): ОС Linux

```
struct task_struct {

    volatile long state;
    void *stack;
    unsigned int flags;

    int prio, static_prio;

    struct sched_entity se;

    struct list_head tasks;

    struct mm_struct *mm, *active_mm;
```

02.7. Блок управління процесом (Process Control Block): ОС Linux

```
    pid_t pid;
    pid_t tgid;

    struct task_struct *real_parent;

    struct list_head children;

    char comm[TASK_COMM_LEN];

    struct thread_struct thread;

    struct files_struct *files;

    ...
};
```

02.8. Блок управління процесом (Process Control Block): ОС Linux

1. ps → список процесів,
приклади: `$ps,  $ps -A,  $ps aux`

2. pstree → дерево процесів (псевдографіка)

3. top → консоль моніторингу та управління процесами

4. kill → завершення процесу (надсилання сигналу процесу),
приклад: `$kill 2929 -9`

03.1. Паралельне виконання обчислювальних процесів

В системах з розділенням в часі (time-sharing system) для організації паралельного виконання обчислювальних процесів використовуються:

1. черга готовності: черга процесів, готових до виконання (ready queue);

2. черга очікування: черга доступу до пристрою вводу/виводу (device queue); кожний пристрій вводу/виводу має свою окрему чергу очікування.

03.2. Паралельне виконання обчислювальних процесів

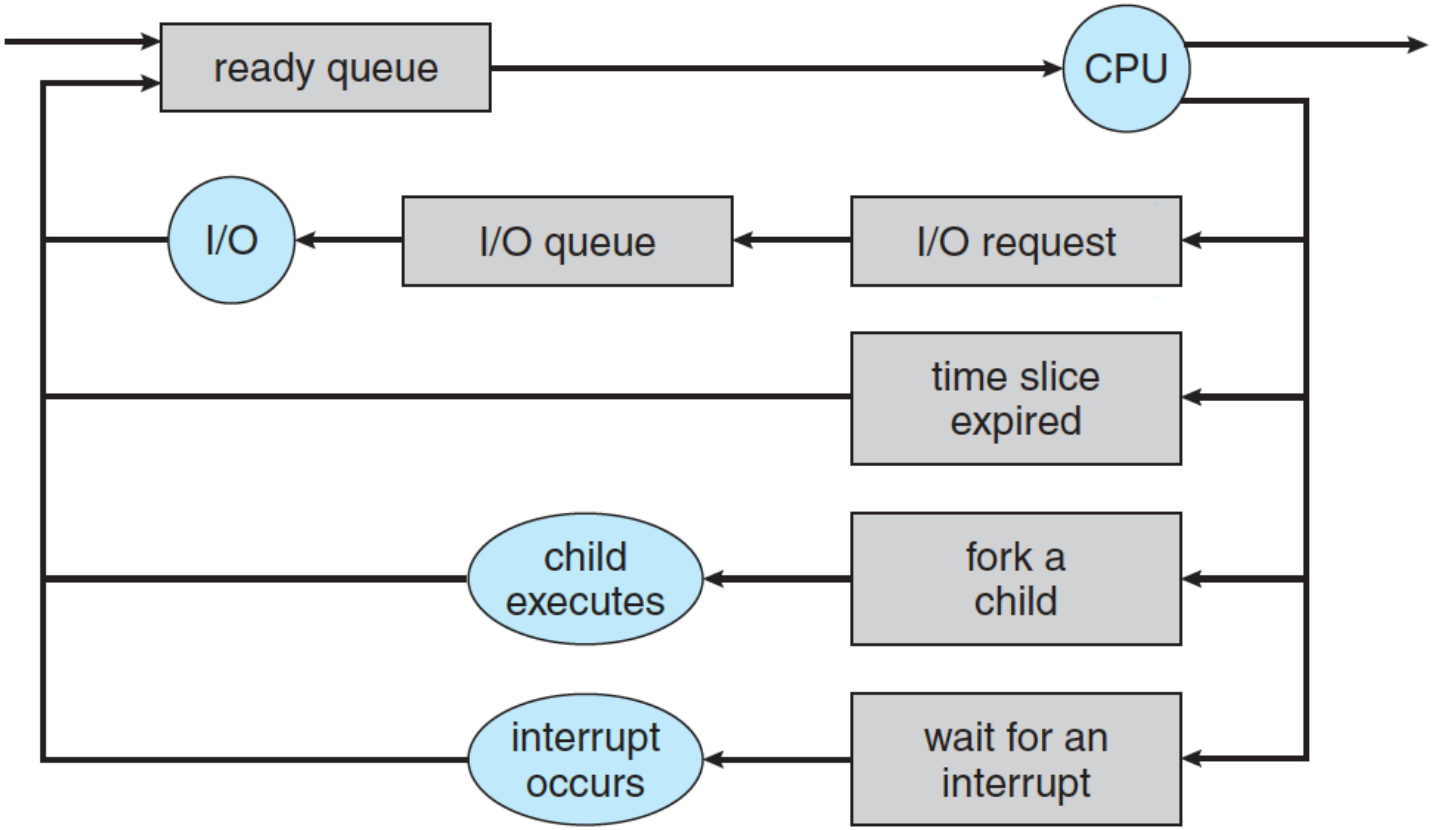


Схема використання черг готовності та очікування

03.3. Паралельне виконання обчислювальних процесів

По часу планування розрізняють:

1. Коротко-термінове планування (short-term scheduler, CPU scheduler); диспетчеризація процесів завантажених в оперативну пам'ять; масштаб часу: менше 100 млс (квант часу (time quantum) процесора = 10 млс, ... 100 млс);

2. Довго-термінове планування (long-term scheduler, job scheduler), завантаження процесів з зовнішньої пам'яті в оперативну пам'ять (скільки паралельних процесів запустити на виконання? → ступінь багатозадачності (degree of multiprogramming)); масштаб часу: більше 1 хв.

03.4. Паралельне виконання обчислювальних процесів

Процеси, які виконуються можна поділити на два класа:

1. I/O bound processes: процеси які більше часу витрачають на очікування та виконання операцій вводу/виводу ніж на обчислення;

2. CPU bound processes: процеси які більше часу витрачають на обчислення, а операції вводу/виводу виконують рідко.

+ У логіці роботи диспетчера крім усього іншого враховується кількісне співвідношення між цими двома класами процесів.

03.5. Паралельне виконання обчислювальних процесів

1.Завданння довготермінового планування: вибрати для завантаження в оперативну пам’ять збалансовану комбінацію процесів обох класів.

2. Якщо кількість процесів 1-го класу в ОП буде набагато переважати кількість процесів 2-го класу, то черга готовності буде майже завжди пустою (у short-term scheduler не буде роботи).

3. В протилежному випадку черга очікування буде майже завжди пустою, а в роботі з пристроями вводу/виводу будуть великі затримки.

03.6. Паралельне виконання обчислювальних процесів

1. Середньо-термінове планування (medium-term scheduler) — це спосіб забезпечити згаданий вище кількісний баланс за рахунок вивантаження деяких процесів з ОП.

2. При цьому зберігається стан вивантажених процесів, що дає можливість їх подальшого заванатаження і продовження виконання.

3. Ця схема роботи отримала назву свопінг (swapping).

03.7. Паралельне виконання обчислювальних процесів

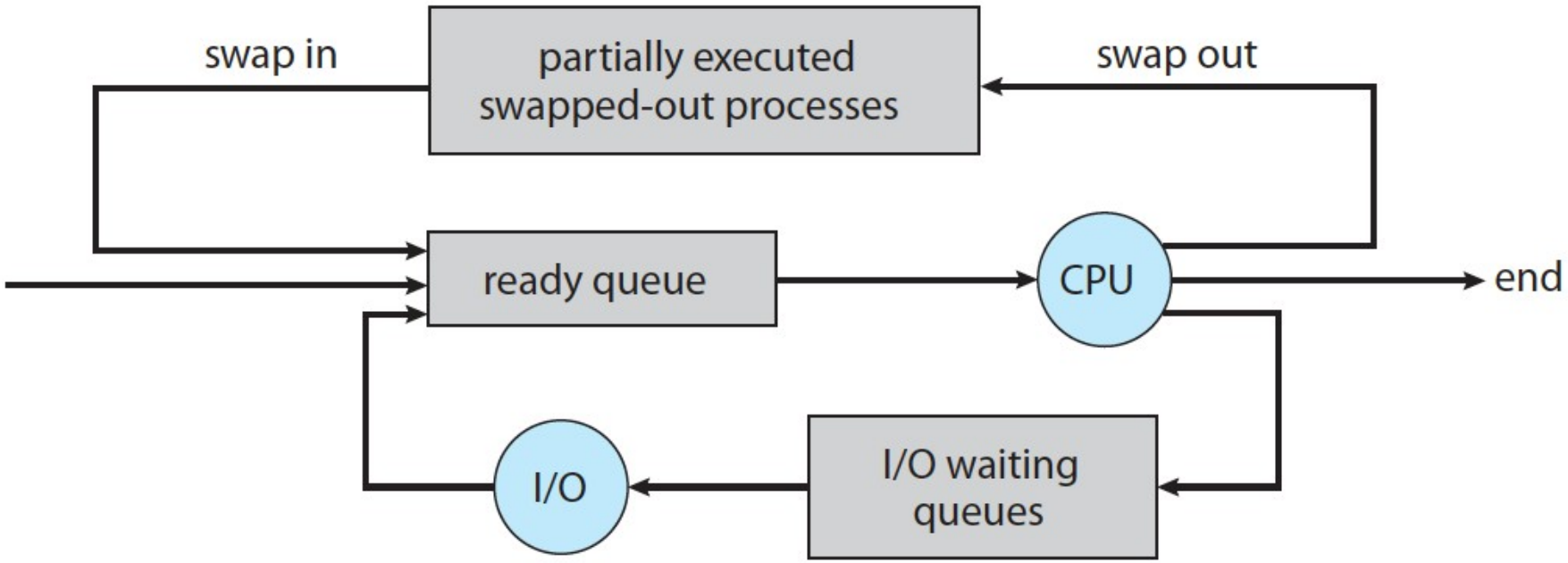


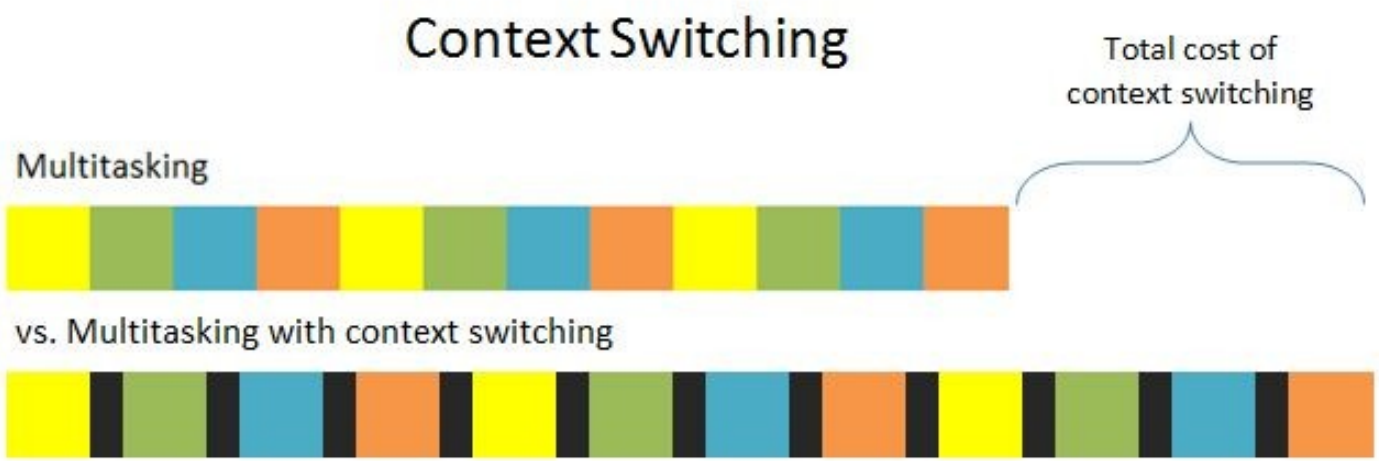
Схема використання середньо-термінового планування

04.1. Переключення контексту процесів (Context Switch)

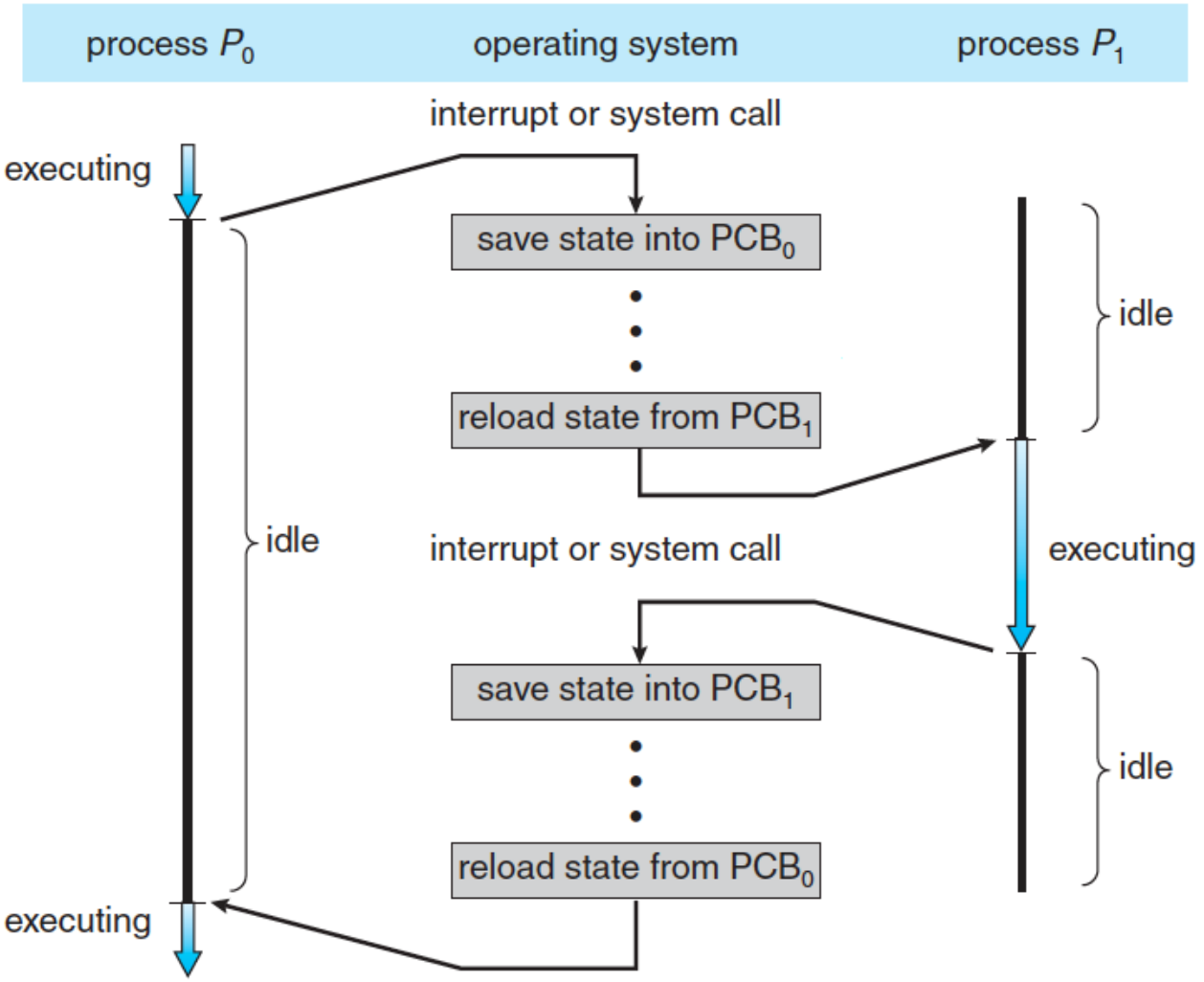
Контекст процесу - це стан регістрів процесора та стан операційного середовища поточного процесу.
Зокрема в ОС UNIX розрізняють:
1) Користувацький контекст (вміст користувацького адресного простору: сегменти коду, даних, стеку + сегмент спільної пам'яті (shared memory) + сегменти файлів відображених у віртуальну пам'ять);
2) Регістровий контекст (вміст апаратних регістрів: лічильник команд, регістр стану процесора, вказівник стека, регістри загального призначення і т.д.);
3) Контекст системного рівня (структури даних ядра ОС, пов'язані з даним процесом: статична та динамічна частина (стеки, які використовуються процесом при його виконанні в режимі ядра), в тому числі дескриптор процесу).

04.2. Переключення контексту процесів (Context Switch)

Для забезпечення багатозадачності з розділенням в часі потрібно виконувати операцію заміни (переключення) контексту (коли попередній процес залишає CPU, а наступний займає його місце).



04.3. Переключення контексту процесів (Context Switch)



04.4. Переключення контексту процесів (Context Switch)

Переключення контексту складається з двох кроків:
1) збереження контексту попереднього процеса;
2) завантаження контексту наступного процеса.

Збереження контексту включає:
1. Збереження регістрового контексту у PCB процеса (в тому числі значення лічильника команд);
2. Збереження динамічної частини системного контекста;
3. Збереження службових даних по управлінню пам'яттю процеса для відновлення його адресного простору (в тому числі вміст translation lookaside buffer (TLB) = буфера асоціативної трансляції адрес віртуальної пам'яті в адреси фізичної пам'яті);

04.5. Переключення контексту процесів (Context Switch)

1. Переключення контексту може бути реалізовано: 1) апаратно; 2) програмно.

2. В сучасних системах переключення контексту в своїй основі програмне з використанням апаратної підтримки (наприклад, в деяких процесорах реалізована можливість збереження всього регістрового контексту одною командою).

3. Перевагою програмного переключення контексту перед апаратним є його селективність: зберігається лише вміст тих регістрів які потрібні, тоді як апаратно зберігається вміст всіх регістрів.

04.6. Переключення контексту процесів (Context Switch)

Втрати у продуктивності роботи («розплата» за багатозадачність):

1. Витрати часу та обчислювальних ресурсів на процедуру переключення конексту.
2. При переключенні контексту відбувається очищення конвеєру команд та даних процесора.
3. Вміст кеша процесора (особливо 1-го рівня), «оптимізований» під попередній процес не підходить для нового процеса.
4. При переключенні на процес, який довго не виконувався, може запуститись підкачка його сторінок пам’яті з ПЗП.

04.7. Переключення контексту процесів (Context Switch)

Оцінка швидкості переключення контексту:

1. «Типова швидкість складає декілько мілісекунд.» «A typical speed is a few milliseconds.» [Silberschatz et al., 2012]

2. За іншими оцінками швидкість переключення контексту складає від 1 до 200 мкс (в середньому менше 10 мкс).

05.1. Запуск та зупинка процесів [на прикладі ОС Linux]

1. Процеси утворюють ієрархію (дерево процесів), в якій кожний процес (окрім самого першого, див. init) має одного «батька» і може мати декілько синівських процесів.

2. fork() : Створення нового процесу в UNIX-подібних ОС (в тому числі в ОС Linux) відбувається шляхом клонування батьківського процесу (створюється копія поточного процесу з новим pid).

05.2. Запуск та зупинка процесів [на прикладі ОС Linux]

3. exec(): В разі необхідності виконавчий код новоствореного процесу замінюється на виконавчий код вказаної програми.

4. wait(): Після створення батьківський та синівський процеси виконуються одночасно. При цьому батьківський процес перед своїм завершенням повинен дочекатися завершення синівського процесу. Завершений синівський процес, батько якого ще не викликав відповідний wait(), в UNIX-подібних ОС називається zombie.

05.3. Запуск та зупинка процесів [на прикладі ОС Linux]

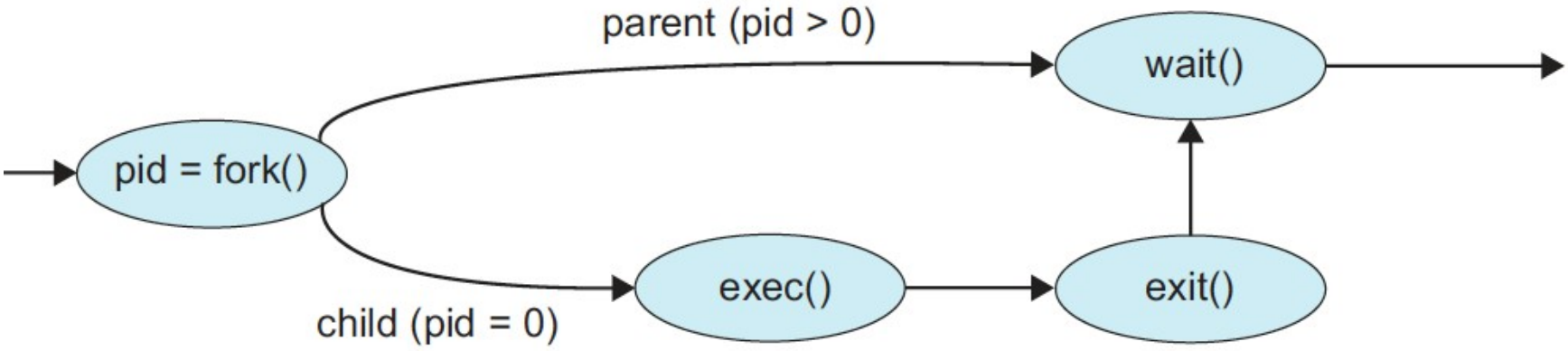


Схема використання системних викликів fork, exec, wait, exit

05.4. Запуск та зупинка процесів [на прикладі ОС Linux]

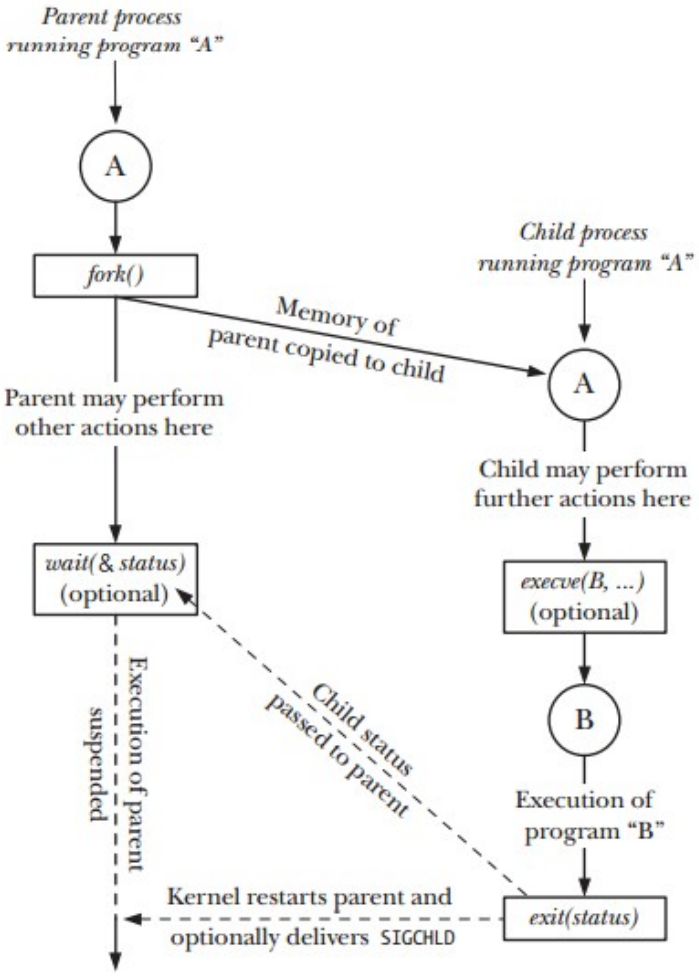


Схема використання системних викликів fork, exec, wait, exit

06. Рекомендована література

1. Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Operating System Concepts, 10th Edition, Wiley, 2018. – 942 p.

2. William Stallings, Operating Systems: Internals and Design Principles, 9th Edition, Pearson, 2017. - 800 p.

3. Andrew S. Tanenbaum, Herbert Bos, Modern Operating Systems, 4th Edition, Pearson, 2014. - 1136 p.

4. Andrew S Tanenbaum, Albert S. Woodhull, Operating Systems Design and Implementation, 3rd Edition, Pearson, 2006. - 1080 p.

5. Remzi Arpaci-Dusseau, Andrea Arpaci-Dusseau, Operating Systems: Three Easy Pieces, CreateSpace Independent Publishing Platform, 2018. - 714 p.