2. Процес та його життєвий цикл.

Екземпляр програми, яка в даний момент виконується на комп'ютері, називають ***процесом***. Поняття процесу має кілька складових. По перше, це потік виконання команд який реалізує алгоритм програми. По друге — ***повторно використовувані ресурси***, що виділені процесу. Це процесорний час, оперативна та віртуальна пам'ять, канали вводу-виводу, відкриті файли та резервовані ресурси зовнішніх пристроїв. По третє — вміст регістрів процесора та значення внутрішніх змінних програми які разом визначають внутрішній стан процесу. Нарешті, це значення зовнішніх відносно процесу змінних, які ідентифікують процес та виділені йому ресурси і фіксують ***стан процесу*** з точки зору ОС. Ці змінні зазвичай є елементами оперативних структур даних ОС. ***Життєвий цикл*** кожного процесу можна відобразити в вигляді діаграми на якій позначені стани процесу та можливі переходи між ними. На рис. 2.1 приведена діаграма станів процесів в гіпотетичній ОС.

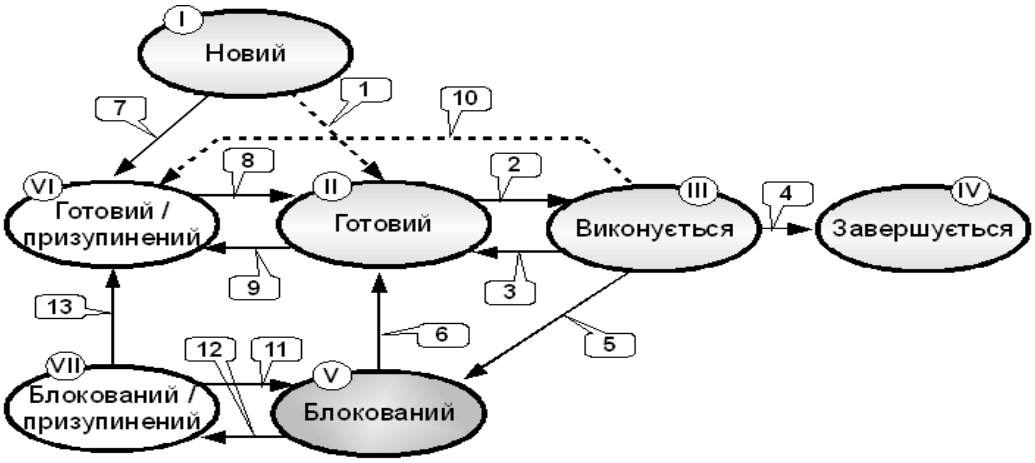

Рис. 2.1. Модель життєвого циклу процесу з сімома станами.

Розглянемо особливості кожного стану. “***Новий”*** процес (I) створюється ОС (можливо за “дорученням” іншого процесу) для виконання певної програми. При цьому ОС створює та розміщує в пам'яті структури даних, що ідентифікують процес, зберігають інформацію про його стан та містять інформацію необхідну для управління процесом. Ці три типи інформаційних структур об'єднуються терміном ***управляючий блок процесу*** (process control block – PCB). Крім того ОС повинна розмістити в пам'яті код програми і дані та виділити пам'ять для стеків які використовуються для викликів процедур. Разом PCB, пам'ять (точніше ***адресний простір*** — список адрес елементів пам'яті від деякої мінімальної до максимальної, до яких процес має доступ) програми, стеки і, можливо, сумісно використовуваний кількома процесами адресний простір складають ***контекст образу процесу***.

Якщо наявних в даний момент ресурсів в системі достатньо для виконання вищезгаданого то процес переходить в стан “***готовий***” до виконання (II). Коли ж ресурсів (в першу чергу пам'яті) недостатньо, процес призупиняється доки не з'явиться можливість їх виділення. При цьому ОС використовує механізм ***свопінгу*** (жарг. “підкачка”) для переміщення образу процесу на диск, а при вивільненні ресурсів — зворотного переміщення в оперативну пам'ять і, таким чином, переводить процес із стану “***готовий / призупинений***” (VI) в “готовий” (переходи 8, 9).

Всі готові до виконання процеси знаходяться в упорядкованій черзі і час від часу переводяться ***диспетчером*** (компонентом ОС що реалізує почергове надання процесам процесорного часу) в стан “***виконується***” (III). При цьому процес отримує в своє розпорядження центральний процесор який виконує програму до завершення виділеного ***кванту часу***, або до запиту процесом обслуговування зі сторони ОС, результати якого не

можуть бути отримані в межах поточного кванту часу. Наприклад, операція вводу даних в програмі на мові програмування високого рівня транслюється компілятором в виклик відповідного ***сервісу*** ОС (***системний виклик***) і наперед ніколи невідомо, коли після початку операції вводу дані будуть реально доступні в програмі. Тому диспетчер вивільняє процесор і переводить процес в стан "***блокований***" (V). В цьому стані процес буде перебувати до приходу ***асинхронного повідомлення*** (час приходу невідомий) про те, що очікувані дані вже доступні всередині ОС і операція вводу може бути завершена. Це повідомлення обробляється ОС яка і переводить процес в стан готового до виконання.

Якщо в блокованому стані знаходиться досить багато процесів і є процеси в стані "готовий / призупинений", то ОС може перерозподілити ресурси таким чином, щоб перевести деякі процеси із стану "готовий / призупинений" в стан "готовий" за рахунок вивантаження на диск частини блокованих процесів. Останні при цьому до моменту настання очікуваної події знаходитимуться в стані "***блокований / призупинений***" (VII).

Оскільки блоковані і блоковані / призупинені процеси очікують настання асинхронних подій — їх черги неупорядковані, тоді як черги готових та готових / призупинених упорядковані. Тип упорядкування для різних черг може бути різним і визначається ***стратегією планування*** (загальним планом діяльності, що охоплює тривалий період часу як спосіб досягнення складної мети, в даному випадку — максимальної продуктивності системи) і диспетчеризації процесів, яку реалізує дана ОС.

Результатом завершення виконання програми є перехід процесу із стану виконання в стан "***завершується***" (IV). Це не обов'язково означає видалення процесу із системи. Часто образ такого процесу містить інформацію необхідну іншим процесам. В цьому випадку частина структур PCB деякий час зберігається і після завершення процесу. В ОС сімейства UNIX такий стан процесу називається ***зомбі*** (zombie).

І ще кілька загальних зауважень до діаграми на Рис. 2.1. На рисунку зображено всі можливі стани процесу і можливі переходи між ними. Одночасно в кожному стані може знаходитись багато процесів, за винятком стану "виконується" в якому може знаходитись не більше процесів ніж мається процесорів в обчислювальній системі. При цьому частину переходів (1, 2, 6 - 13) ініціює ядро ОС незалежно від тексту програми, що виконується. Перехід (3) як правило примусово виконує диспетчер при закінченні виділеного процесу кванту часу. Це так звана ***витісняюча багатозадачність*** (preemptive multitasking) що реалізована в більшості сучасних ОС загального призначення. При ***кооперативній*** моделі багатозадачності (cooperative multitasking) процес сам ініціює перехід (3), виконуючи системний виклик звільнення процесора. Перехід (4) відповідає виконанню системного виклику завершення процесу. Він може бути вказаний в програмі явно як виклик процедури або неявно по досягненню кінця програми. Можливий також варіант примусового завершення процесу іншим процесом який має відповідні права.