역사-의미

1. 사용자-프로그램-실행(프로세스)
2. 사용자 증가
   사용자_1, 사용자_2, 사용자_3
   순서대로 처리 - 순차 처리(Sequence Processing)
   [문제] 각 사용자의 프로그램 실행(컴파일, 로드, 런)을 위하여 준비하는데
      사용자 마다 특성(언어)이 다름(C 언어 프로그램 실행을 위한 준비 후 실행)
   같은 특성의 프로그램을 모아서 실행 - 일괄 처리(Batch Processing)
   [장점] 기계(컴퓨터) 사용의 효율성 증대 [단점] 불편
3. 컴퓨터 메인 메모리 증가
   여러 사용자의 프로그램을 메모리에 적재하여,
      각 사용자의 준비단계를 줄임 - 멀티프로그래밍(Multi-Programming)
4. 컴퓨터의 응답성(대화성-Conversional, 개입성-Interactiveness) 요구
   여러 사용자의 프로그램 실행을 위하여 시간을 나누어서, 각 다른 시간 구간에,
      다른 사용자의 프로그램을 실행 - 시분할 시스템(Time Sharing System)
5. 각 사용자의 프로그램 개수가 증가
   하지만 하나의 프로세서(Processor, CPU, 처리기)만 이용 - Single Processing
   시스템에 존재하는 실행 프로그램(프로세스, Process)은 하나뿐임
6. 시스템의 프로세서 개수가 증가
   하나의 프로그램을 하나의 프로세서에게 맛김(실행)으로
      시스템에서는 여러 개의 프로그램이 동시에 실행됨 - Multi-Processing
      (동시처리-Concurrent Processing, 병렬처리-Parallel Processing)
7. 네트워크(통신)의 성능 향상
   여러 개의 프로세스가 동작하는 시스템이
   여러 개가 있어(다른 컴퓨터 시스템, 지역적으로 떨어질 수도 있음)
   하나(어떤)의 컴퓨터의 여러 프로세스를
   다른 컴퓨터 시스템으로 옮겨 실행 - Distributed Processing

*. 장애허용 시스템(결함 감내 시스템, Fault Tolerant System)
*. 실시간 처리 시스템(Real Time System)