



Anexo: evolución

☰ Materia	ISO Teoría
⚙ Estado	Done
📅 Fecha	

Evolución de un SO

1. Procesamiento en serie
2. Sistemas por Lotes Sencillos (batch)
3. La multiprogramación
4. Tiempo compartido

Evolución de un SO

Un SO evoluciona con el objetivo de:

- soportar nuevos tipos de HW
- brindar nuevos servicios
- ofrecer mejoras y alternativas a problemas existentes en la planificación y en el manejo de memoria.

1. Procesamiento en serie

No existía un sistema operativo. Las máquinas eran usadas desde una consola que contenía luces, interruptores y dispositivos de entrada e impresoras.

La planificación era complicada y requería alto nivel de especialización y costos. La configuración de las máquinas era complicada, había que cargar el compilador, la fuente, salvar el programa compilado, cargarlo y linkearlo.

2. Sistemas por Lotes Sencillos (batch)

El software controla la secuencia de eventos, los trabajos se colocan juntos y los programas vuelve un monitor cuándo finalizan su ejecución. No hay interacción con el usuario mientras se ejecutan los trabajos.

Es la época de las máquinas grandes y las *tarjetas perforadas* (cosas super piolas que recomiendo buscar en wikipedia)

Hay una baja utilización de la CPU, los dispositivos de E/S son mucho más lentos con respecto a la misma y, ante las instrucciones e/s el procesador permanece ocioso. Cuando se completa la instrucción de E/S se continúa con la ejecución del programa que se estaba ejecutando.

Una solución a esto fue:

3. La multiprogramación

La operación de los **sistemas de batch** se vio beneficiada del *spooling* de las tareas: solapar la E/S de una tarea de la ejecución de otra. Las tareas, cargadas en disco, no necesitaban ejecutarse en el orden en que fuesen cargadas y el sistema operativo era capaz de mantener varias tareas en memoria al mismo tiempo.

La secuencia en que los programas se ejecutaban era de acuerdo a prioridad u orden de llegada. Cuando el proceso necesitaba realizar una operación de E/S la CPU, en lugar de permanecer ociosa, era utilizada para otro proceso. Después que se completa la atención de la interrupción, el control puede o no retornar al programa que se estaba ejecutando al momento de interrupción.

4. Tiempo compartido

- Utilizar la multiprogramación para manejar múltiples trabajos interactivos
- El tiempo del procesador es compartido entre múltiples trabajos.
- Los usuarios pueden acceder **simultáneamente** al sistema utilizando terminales
- Los procesos usan la CPU por un periodo máximo de tiempo, luego del cuál se le da la CPU a otro proceso.