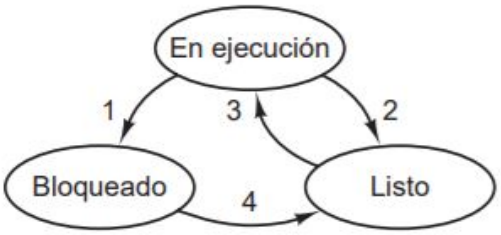
Taller el Viernes 13 de marzo, traer laptop.

Servicios de un Sistema Operativo

Estados de un proceso

- Cada proceso es independiente (se debe cumplir la garantía, no permitir que otro proceso ingrese al espacio de direccionamiento de otro proceso), posee su propio contador de programa y estado interno.
- ¿Para qué un proceso necesita comunicación con otro?
    - Para compartir información.
    - Dos formas: Memoria compartida y por mensajes (cliente-servidor).
- Existen dos modelos:
    - Simplificado.
    - Completo.
- Ejemplo: cat cap1 cap2 cap3 | grep árbol
    - *grep* espera el insumo del *cat*, o sea, un proceso puede generar una salida que sea la entrada de otro proceso; si *grep* está listo para ejecutarse pero falta la entrada, entonces permanece bloqueado..
    - Premisa: No se puede determinar la duración exacto de un proceso; no es reproducible. En este caso la concatenación de los 3 caps no va a durar siempre lo mismo.
- Nota: *mmap* para dividir memoria y establecer bloques en C.

Modelo simplificado de los estados de un proceso



Transiciones

1. El proceso se bloquea para recibir entrada.
2. El planificador selecciona otro proceso.
3. El planificador selecciona este proceso.
4. La entrada ya está disponible.

Estados

- **En ejecución:** El proceso está en el procesador. Cuando se pasa un proceso a listo, este estado se libera, entonces procede con ingresar otro proceso que esté listo.
- **Bloqueado:** Cuando ocupa un insumo, espera aquí. Cuando esté listo pasa hasta el estado “Listo”.
- **Listo:** Sino hubo algún error o se acabó el tiempo de ejecución, se ingresa en este estado.
- **¿Puede existir una transición de “Bloqueado” a “En ejecución”?**
    - **NO, porque le pasa por encima a la calendarización del S, ya que quita el proceso en “En ejecución” sin importar nada.**

- ¿Qué pasa si se encicla un proceso? Debe haber un calendarizador o planificador para terminar dicho proceso.

Modelo 5 Estados (Completo)

Estados

- **Nuevo:** Un proceso que se acaba de crear, pero no está admitido en el grupo de procesos ejecutables. Pero, si se reserva la memoria y los registros de frontera están listos.
- **Listo:** Se ingresar al procesador para ser ejecutado en cualquier momento.
- **Bloqueado:** Un proceso que no se puede ejecutar por un evento determinado (al esperar un dato en memoria, interrupción, etc.) o terminar los datos de entrada.
- **Ejecutando:** Proceso que está en el CPU en un instante. Si ocurre un error, entonces pasa a “Saliente”.
- **Saliente:** Se libera del grupo de procesos ejecutables. Si pasa alguna causa de terminar el proceso (salida normal, error fatal, error por usuario y eliminado por otro proceso) se coloca en este estado.
- **¿Puede existir una transición de “Bloqueado” a “En ejecución”?**
    - **Mismo caso que en el modelo simplificado.**
    - **NO, porque le pasa por encima a la calendarización del S, ya que quita el proceso en “En ejecución” sin importar nada.**

- Nota: El quantum es el adecuado, si es grande durará mucho en ejecutar los programas y si es pequeño, no tiene la capacidad.
    - Es de 100ms, usualmente.

Transiciones

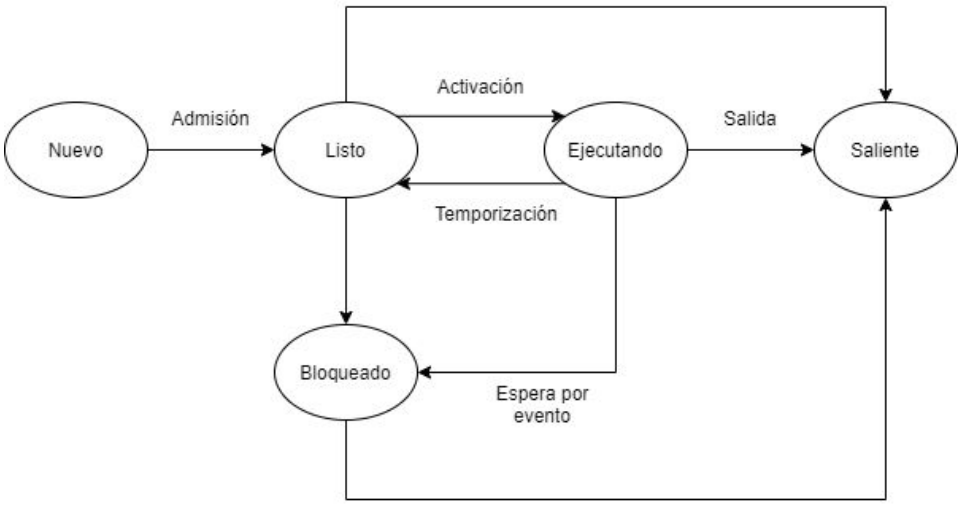
- **Transición de null a nuevo**
  - Se crea un nuevo proceso (arranque del sistema, una llamada al sistema, petición de usuario, trabajo por lotes; todas vistas anteriormente en clases) para ejecutar un programa.
- **Transición de nuevo a listo (Admisión)**
  - El SO realiza esta transición cuando ya se encuentre preparado para ejecuta un nuevo proceso (cuando ya está mapeado en memoria).
  - Deben estar listo los contadores de programas, tablas, direcciones, registros de frontera y todos los elementos necesarios. Si no olvidan esto, es digno de enseñarselo al profe Beto.
  - Generalmente se tiene una política para una cantidad máxima de procesos. Pueden haber SO que permiten 3 procesos. También depende de la arquitectura del equipo, pueden haber programas que exijan más espacio en memoria y por ende, no puede ser ejecutado o si pero de manera ineficiente.
  - ¿Qué ocurre si no existiera una cantidad máxima de procesos?
    - Se llena la memoria, entonces un nuevo proceso bota otro y así sucesivamente.
- **Transición de listo a ejecutando (Activación)**
  - El SO selecciona un proceso que se encuentre en el estado “Listo” para ser ejecutado, según las reglas del planificador (dependiendo del algoritmo).
- **Transición de ejecutando a saliente (Salida)**
  - El proceso en ejecución se finaliza por parte del SO, tanto si el proceso terminó con éxito o por un error.
- **Transición de ejecutando a listo (Temporización)**
  - La única y principal razón es que se le acabó el tiempo máximo de estar en el procesador y debe ceder el campo a otro proceso (esto lo determina el calendarizador).
- **Transición de ejecutando a bloqueado (Espera por evento)**
  - El proceso solicita algún recurso (datos, hardware, I/O) por el cual debe esperar, sino estaría en el CPU ocupando espacio innecesariamente.
  - Generalmente se realiza por medio de una llamada al sistema (como el *read* en C).
- **Transición de bloqueado a listo (Sucede evento)**
  - Cuando ocurre el evento por el cual estaba esperando.
  - ¿Donde se coloca la cola de listos?
    - Depende del calendarizador.
- ***Transición de listo a saliente**
  - La implementan algunos SO.

- Es cuando el padre puede terminar el proceso del hijo en cualquier momento, por ejemplo, esto puede suceder cuando se usa *fork* y se crea un proceso “hijo”.

- ***Transición de bloqueado a saliente**
  - Se da si por alguna razón se debe finalizar el proceso, aplica el mismo concepto que la transición anterior.

- Nota: Si pide modelo completo con las dos alternativas, se hacen las dos transiciones con *. Si solo pide el modelo completo, se realiza sin ellas.

Modelo completo con las dos alternativas



Cambio de proceso

- “Al final de cuentas es solo decidir cuál proceso sigue, ponerlo a ejecutar y listo”, pero:

- ¿Que se debe tomar en cuenta (variables de entorno) para un cambio de contexto?
    - Puede generar overhead al estar cambiando de contexto al consumir mucho tiempo del procesador, cambiar registros, guardar el PC, punteros, registros de frontera y demás.

Datos generales de un proceso (Esto es por cada proceso, todo está en memoria, o sea, es volátil)

- **Identificador**: No puede faltar.
- **Estado**: Saliente, bloqueado, etc.
- **Prioridad**: La decide el SO por medio del planificador.
- **PC**: Saber por donde va el proceso, porque sino volver a ejecutar todo.
- **Punteros a memoria:** Son los registros y los system call.
- **Datos de contexto:** Todos los datos cuando se quita el proceso.
- **Información de auditoría:** Tiempo de ejecución, archivos abiertos, datos.
    - Principio de localidad: si un programa ingresa a una dirección, existe una alta posibilidad de que ingrese a direcciones cercanas.
- **Información de E/S:** Quien introduce datos, cual sector o bloque del disco está escribiendo o leyendo. Guarda la dirección, por ejemplo, cuando lee y necesita detener el proceso, y así, no volver a empezar desde cero cuando retome el proceso.

Bloque de control de proceso (PCB)

- El PCB se guarda en memoria principal.
- El SO también tiene su propio PCB.
- Es una de las estructuras más importantes del SO.
- Contiene toda la información sobre un proceso que necesita saber el SO y en el cambio de contexto, esto es lo esencial para no perder por donde iba.

- Para realizar el cambio de contexto, necesita una tabla de PCB para tener el control de todos los PCB en ejecución.
- ¿Que contiene el PCB?
    - Contador del programa.
    - Estado del proceso.
    - Apuntadores de pila.
    - Asignación de memoria.
    - Información sobre archivos abiertos.
    - Información de contabilidad y planificación.
    - Identificadores y demás elementos necesarios.