**Programa**: conjunto de instruções descrevendo como a tarefa irá ser realizada.

* Precisa de ser executado

**Processo**: entidade que representa o programa a ser executado

* Representa uma atividade
* Caracterizado pelo código e pela data do programa associado
* Caracterizado pelos valores atuais dos registos internos do processador
* Caracterizado pelo input e output da data que está a ser transferida pelos devices
* Caracterizado pelo seu estado de execução.
* Diferentes processos podem ser executados no mesmo programa
* Existem mais processos que processadores (multiprogramming)

**Modelo do processo:** Num ambiente multiprogramado, a atividade do processador, devido ao switch entre processos, é difícil de se notar.

* É melhor assumir que existe um processador virtual para cada processo.
* Desligar um processador virtual e ligar outro corresponde a fazer o process switching.
* Numero de processadores virtuais ativos <= numero de processadores reais.!!!!
* Alterar entre processos (virtual processors) pode ocorrer por diversas razões não controladas pelo programa (mas também pode ocorrer pelo programa em si) que está a correr e esse programa/processo não pode ser afetado pelo instante (ou local do código) em que está a ser interrompido (switch) e não lhe pode ser restrito o numero de execuções totais ou parciais do processo.

**Diagrama de estados de um processo:**

As transições entre estados podem ser triggered pelo processo em si ou por meios externos, quem handle essas transições é o scheduler (escalonador).

A memoria principal é finita, o que limita os processos existentes.

A melhor maneira de controlar isto é usar uma área secundária (swap área) que pode ser o disco da partição ou um ficheiro.

Um processo que está em run pode ser swapped out para dar memoria a outros processos

O processo depois pode ser swapped in depois da memoria ficar available.

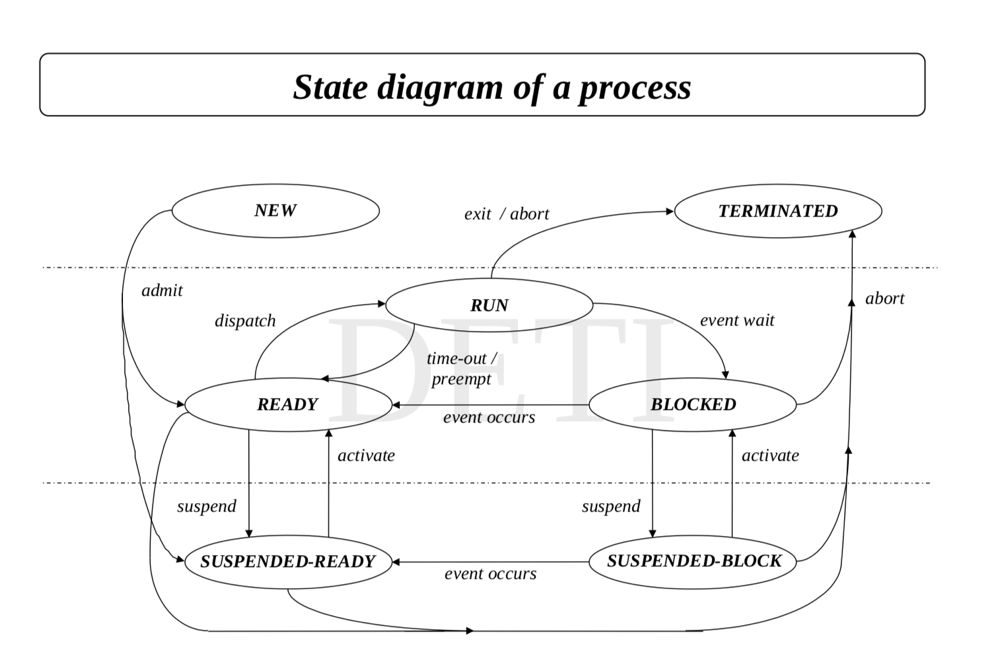
Os processos são criados, existem e depois são acabados.

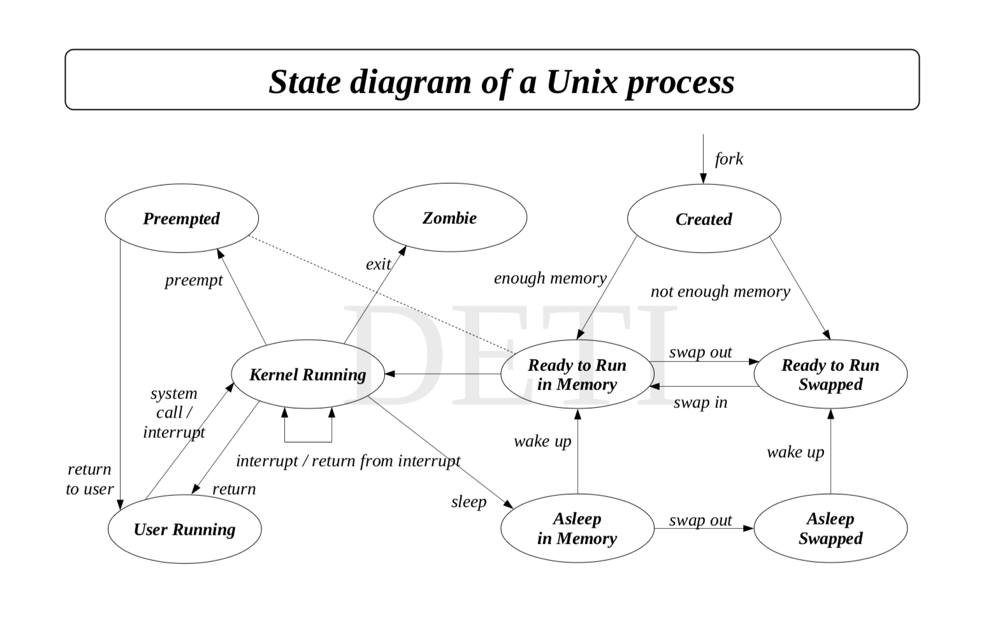
Estados:

* Run: está na posse do processador (running)
* Blocked: está a espera de um evento externo (espera por um acesso a uma resource, a espera que uma operação de IO acabe etc.....).
* Ready: está pronto para run mas esta a espera que o processador fique disponível para começar / Continuar onde estava.
* Suspended-Blocked: o processo foi para a o swap (swapped out) e foi posto a espera de um evento externo ocorra.
* Suspended-Ready: o processo está no swap mas pronto para ser posto a funcionar
* New: o processo é criado mas não admitido como processo executável (a data strcuture do processo é inicializada).
* Terminated: O Processo é mandado considerado como processo não executável (algumas ações podem ainda ser precisas antes de o processo ser descartado).

Eventos:

* Dispatch: um processo que está ready é selecionado para dar run no processador
* Event wait: um processo que estava no run é impedido de prosseguir / continuar, ficando a aguardar até a ocorrência de um evento externo.
* Event Occurs: o evento externo acontece e o processo que estava blocked vai para a fase ready para se preparar e esperar para ir para o processador
* **Time-Out**: o slot de tempo dado ao processo que estava a correr acaba e o processo é removido ao processador
* Preempt: um processo mais prioritário está a espera de correr então o processo é removido ao processador.
* **Suspend**: faz swapped out ao processo
* Activate: fsaz swapped in ao processo
* Admit: o processo é admitido como processo executável
* Exit: o processo indica ao OS que está completo
* abort: o processo é forçado por causa de um erro ou um processo automizado disse para parar a execução.





* Existe em unix dois run states: o kernel running que roda o programa como supervisor e o user running que roda o programa como user normal.
* Ready to run in memory e preempted são equivalentes
* Quando um user process deixa o modo root, pode ser mandado para preempted (por causa de uma instrução mais prioritária que eleque esteja no ready to run in memory).
* Na prática os processos em ready to run in memory e preempted partilham a mesma queue, então são tratados como igual.
* A timeOut transition é igual ao preempt aqui no UNIX.
* Tradicionalmente, a execução de um processo em supervisor mode não pode ser interrompida em UNIX (não deixa real time processing), mas em versões atuais (chamadas de SVR4), o problema foi resolvido dividindo o código em uma sucessão de regiões atómicas de modo a que as internal data structures estão num safes state (estado seguro), o que permite que a execução seja interrompida. Isto corresponde a uma transição entre preempted e kernel running stages