PCS 3446 – Sistemas Operacionais

Aula 25 – CTSS – The compatible Time Sharing System

Madnick & Donovan, Operating Systems, 1974

Introdução

- O CTSS (Compatible Time Sharing System) foi apresentado pela primeira vez de forma rudimentar em 1961, veio a ser usado extensivamente em 1963 e foi descontinuado em 1973.
- Seu estudo se justifica por sua simplicidade e por sua importância histórica, como um dos sistemas time sharing pioneiros, do qual derivaram diversos outros sistemas mais recentes.
- O CTSS incorporou importantes técnicas de administração de recursos, algumas inéditas, e seu modelo abstrato é bastante flexível para ser implementado em diferentes máquinas.

Características

- CTSS foi construído em um hardware IBM 7094 modificado
- Seu uso normal permite até 30 usuários em modo time sharing, e um batch stream controlado pelo Fortran Monitor System (FMS)
- Programas que são executáveis no FMS são executáveis também no CTSS
- Cada usuário em time sharing pode criar e armazenar arquivos, causar a execução desses arquivos e de programas de sistema
- O usuário dispõe de uma variedade de editores de linha, de editores contextuais, e de macro-editores

Usos do CTSS

- CTSS foi usado para desenvolver boa parte do sistema MULTICS e desempenhou um papel importante no desenvolvimento do antigo sistema CP/CMS
- Ajudou a comprovar a eficácia da ideia de uma fábrica de software (baseada em ferramentas de desenvolvimento) com a finalidade de reduzir tempos de desenvolvimento de software, aumentar a qualidade da documentação dos programas desenvolvidos e a eficiência do código

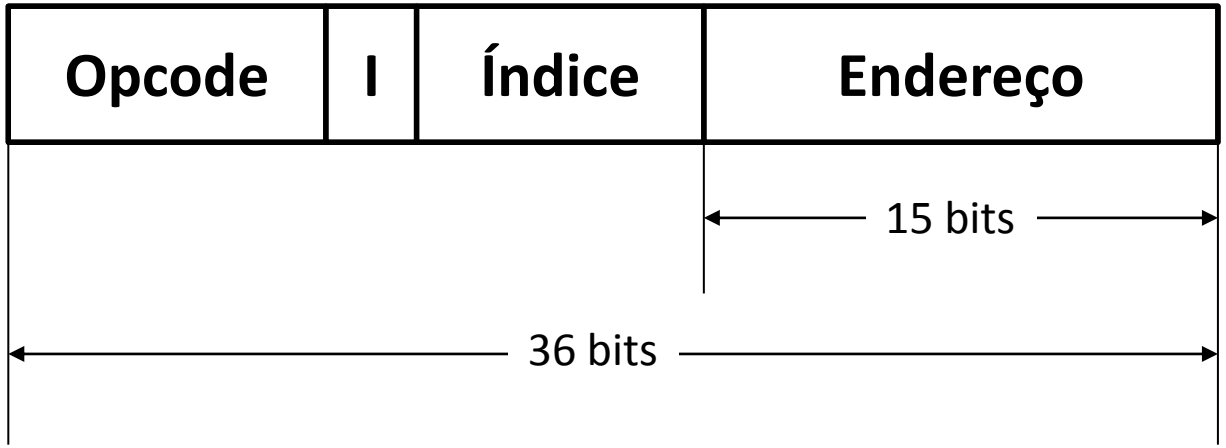
Sessão em terminal de timesharing

- As sessões de timesharing ocorriam em terminais parecidos com máquinas de escrever elétricas ou teletipos, com teclado e impressora de caracteres
- O usuário e o sistema travam um diálogo em que se alternam o usuário (digitando comandos em letras minúsculas), e o sistema) gerando saídas como resposta, em letras maiúsculas)
- Em uma sessão típica, o usuário chama um editor, entra com um texto fonte, armazena-o em arquivo, chama um compilador para processar esse arquivo, edita novamente o arquivo para corrigir erros ou efetuar alterações, compila novamente o arquivo, chama os programas de sistema (loader, linker) para construir um programa executável, manda executar o código assim construído, fornece dados para o programa produzido e encerra a sessão

O sistema IBM 7094

Formato das instruções

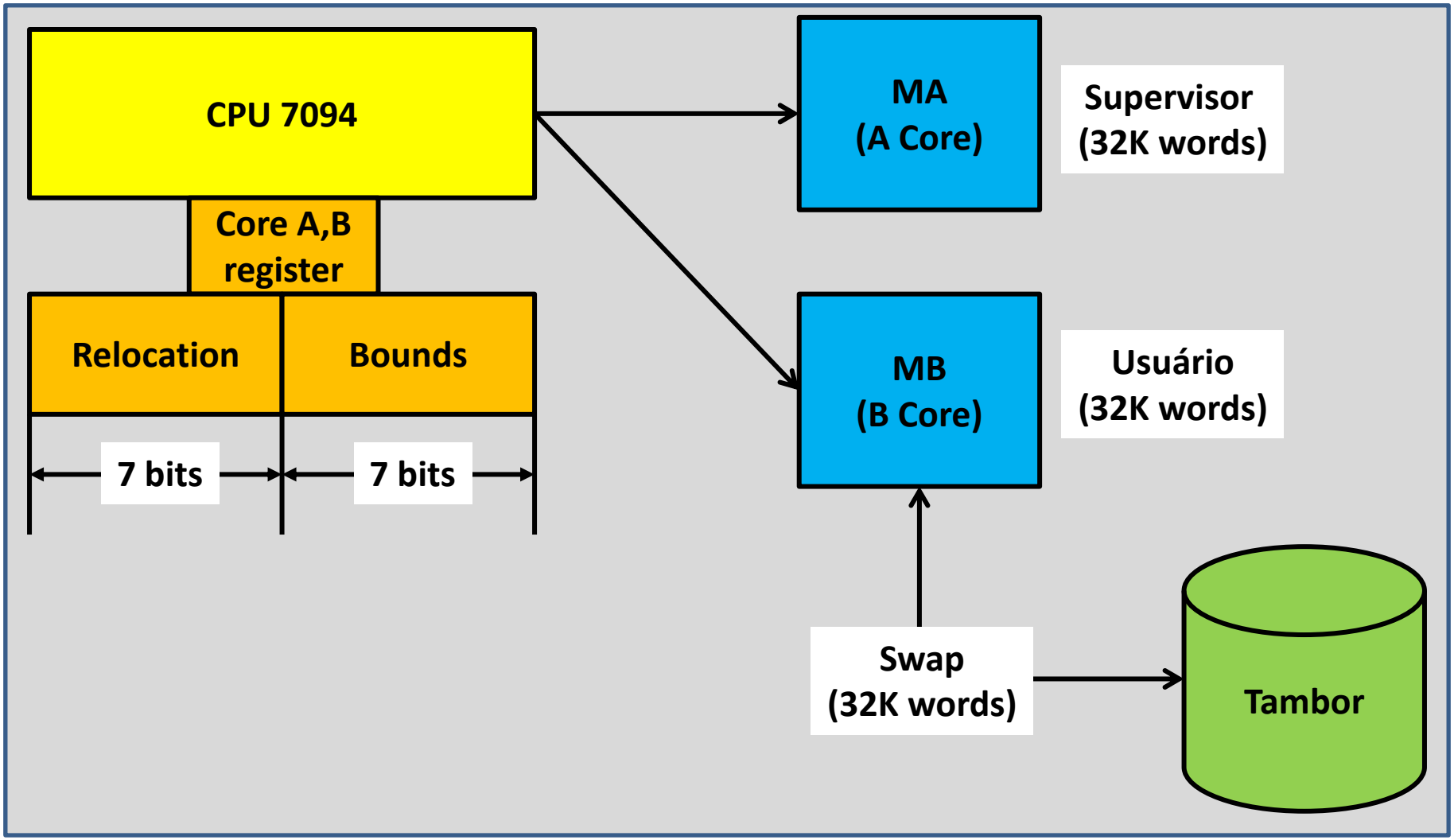
- O CTSS foi construído para ser usado em processadores IBM 7094 modificados
- O tamanho da memória do 7094 era de 32K words de 36 bits cada
- Uma instrução típica ocupava 36 bits, dos quais 15 bits representavam um endereço de memória:



Hardware

- A modificação mais importante foi a da inclusão de um banco adicional de 32K de memória
- Para dar acesso aos 64K resultantes, foi acrescentado um registrador de 1 bit para indicar acesso ao banco A ou ao banco B de memória
- O banco A era usado pelo sistema operacional, e o banco B, pelo usuário
- Dois registradores de 7 bits foram acrescentados para base de relocação dinâmica e para limite de partição
- Um tambor magnético era usado para swapping do banco B em 250ms

Hardware 7094 modificado para CTSS

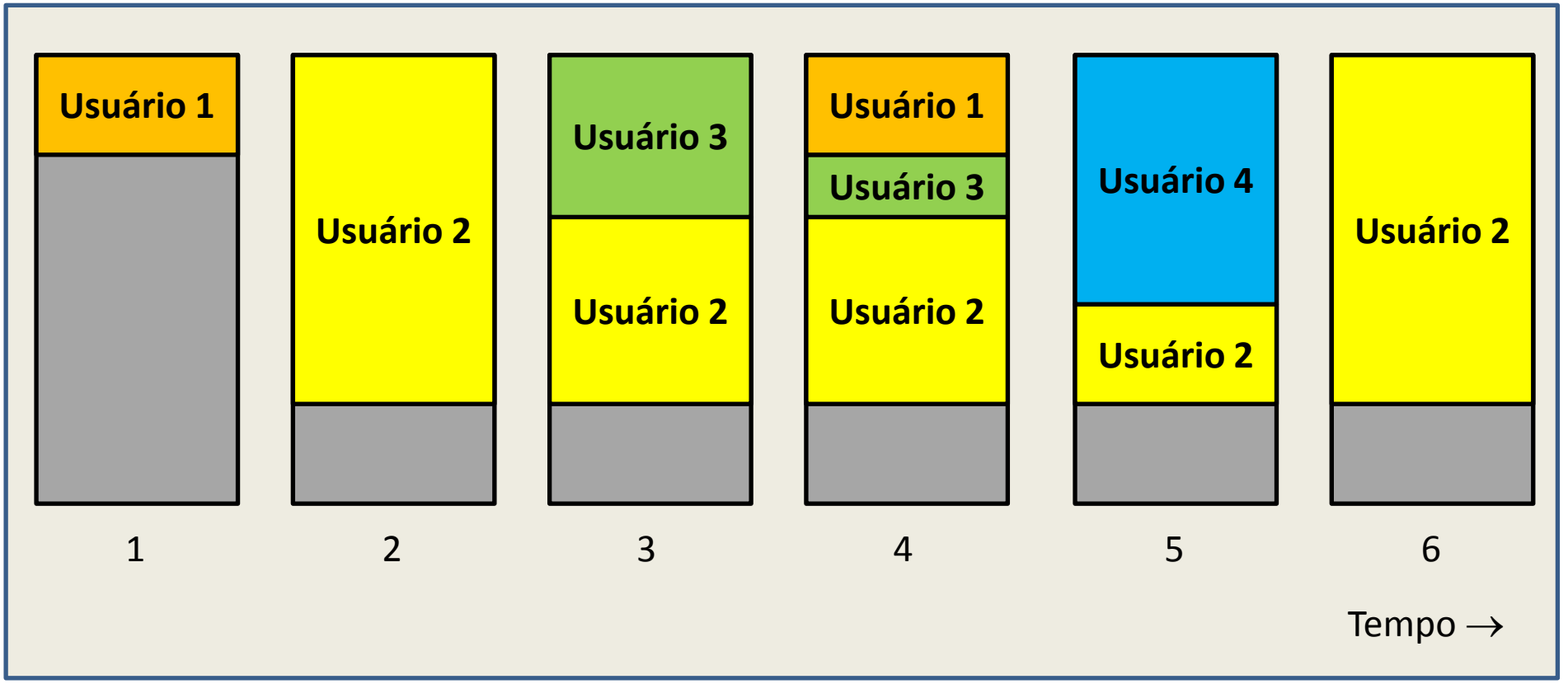


Gerenciamento de memória

- Gerenciamento básico de memória que o ctss usa é um “algoritmo de casca de cebola”, ou seja, baseado em pilha, para o swapping
- Ao usuário é aocada a memória conforme sua necessidade, começando na posição zero do banco de memória B
- Sua necessidade inicial de espaço em memória é determinada pela pelo tamanho do seu programa
- Memória adicional pode ser requisitada durante a execução, para tabelas e para áreas de dados
- A qualquer momento cada usuário tem seu próprio limite de memória, dado pelo endereço mais alto de memória por ele utilizado
- Quando outro usuário for escolhido para utilizar o processador os programas e dados do usuário corrente são devolvidos ao disco
- Para minimizar o tempo de swapping somente a área de memória de fato ocupada é envolvida na operação de swapping, para liberar rapidamente o espaço na memória, para uso do próximo usuário

- A figura que aparece no próximo slide mostra o comportamento de quatro usuários em um certo período de tempo.
- Note-se que, quando o terceiro usuário foi trazido do disco para a memória, não foi necessário remover toda a área do usuário 2
- Portanto quando o usuário 2 foi subsequentemente trazido de volta à memória o seu espaço de endereçamento estará em parte ainda presente, economizando assim o tempo de recarga
- No instante 5, para que se possa trazer para a memória o usuário 4, o usuário 1 precisou ser removido, juntamente com as porções residuais dos usuários 3 e 2

Algoritmo casca de cebola



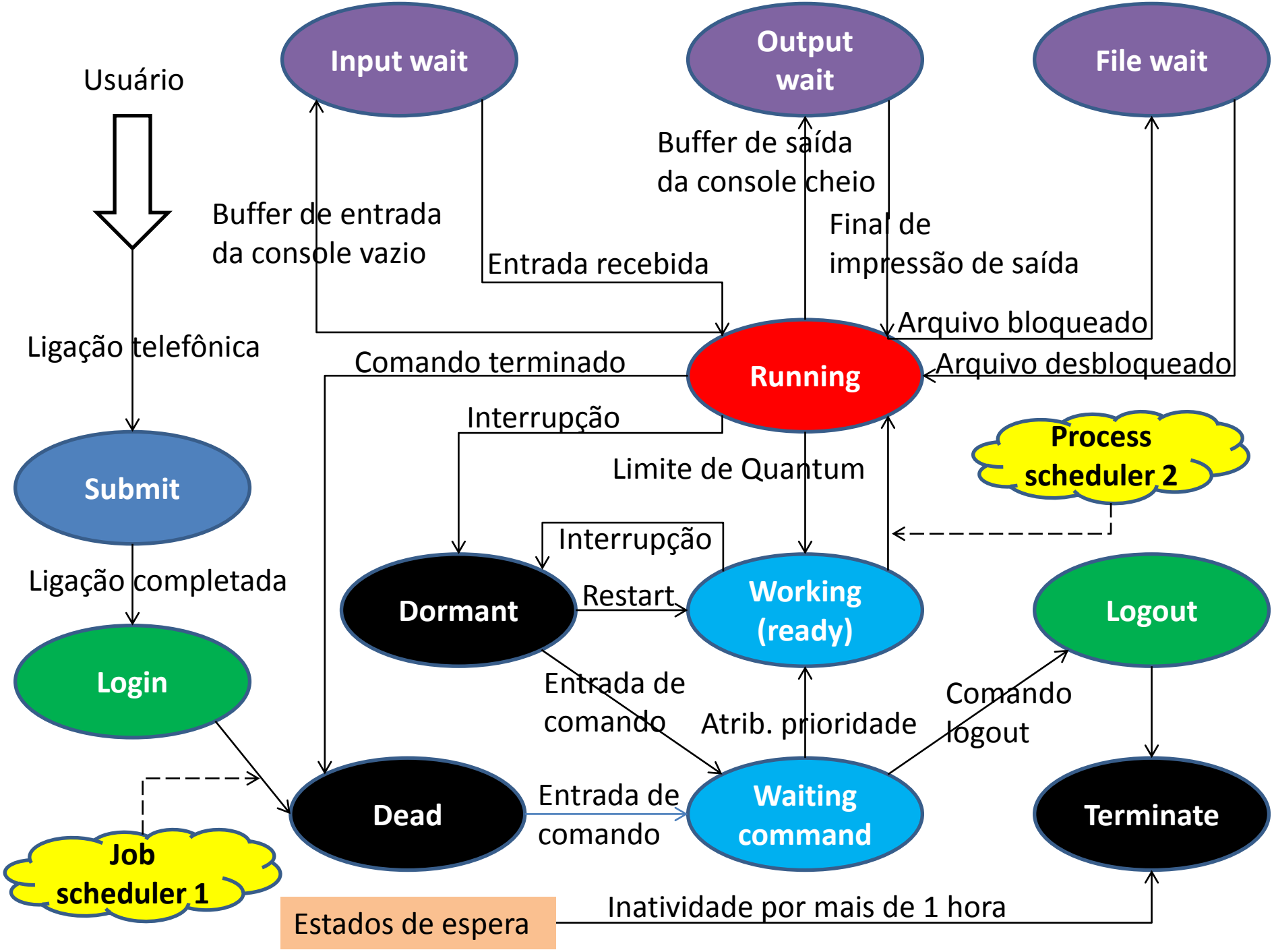
Registradores de relocação e limite

- No CTSS todos os jobs começam no endereço 0 do bloco B de memória
- Por isso, o registrador de relocação nunca é usado
- Poderia ter sido empregado para implementar multiprogramação, ou então para sobrepor os tempos de swap-in e de execução, mas esses recursos não foram implementados.
- Como é utilizado o algoritmo casca de cebola, o registrador de limite permanece necessário para garantir proteção aos fragmentos de outros jobs que continuam presentes na memória e convivem com o job corrente durante a execução deste

Administrador de processador

- Um usuário recebe o processador em quatro possíveis situações:
  a) Final de time-slice
  b) Erro do usuário
  c) Final de execução do job
  d) Entrada/saída na console
- Notar que o usuário não pode ser desalojado da memória enquanto não completar toda a sua atividade de entrada/saída em curso (exceto E/S na console)
- A cada 200 ms o CTSS recupera o controle do processador para determinar se a E/S de console foi completada, ou então se o processador deve ser cedido a outro usuário

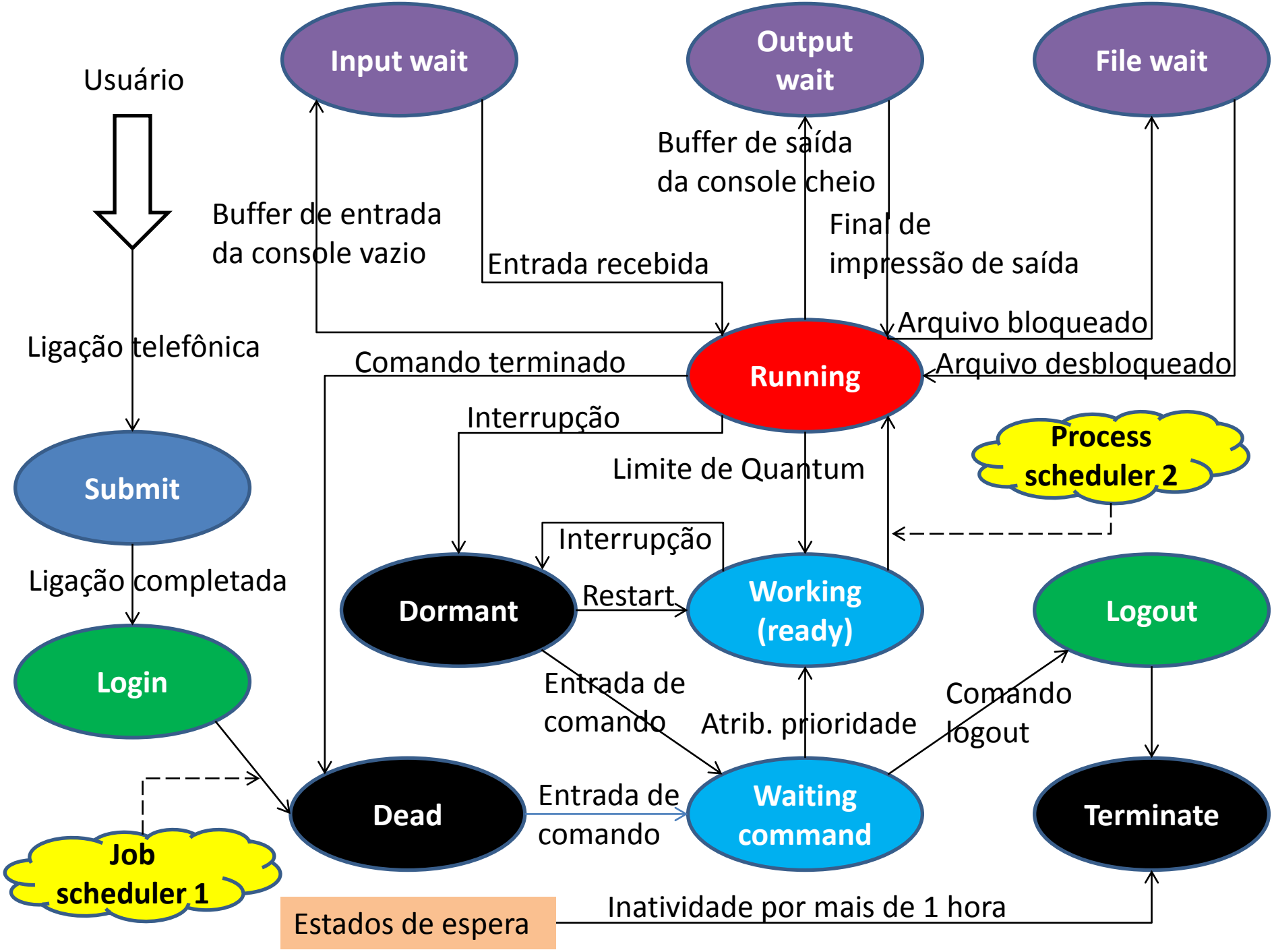
- O modelo da figura (diagrama de estado) mostra os seguintes possíveis estados do processo
  - Submeter – corresponde a um usuário que chegou ao sistema estabelecer uma conexão telefônica com o computador
  - Login – o usuário acabou de completar a ligação telefônica mas a ele ainda não foi alocado nenhum recurso
  - Em operação – corresponde a um estado “pronto” no qual o processo já seja capaz de ser executado desde que seja trazido para a memória e a ele, alocado um processador


Usuário
Input wait
Output wait
File wait
Buffer de entrada da console vazio
Entrada recebida
Buffer de saída da console cheio
Final de impressão de saída
Arquivo bloqueado
Arquivo desbloqueado
Ligação telefônica
Comando terminado
Running
Interrupção
Limite de Quantum
Process scheduler 2
Submit
Interrupção
Ligação completada
Dormant
Restart
Working (ready)
Logout
Login
Entrada de comando
Atrib. prioridade
Comando logout
Dead
Entrada de comando
Waiting command
Terminate
Job scheduler 1
Estados de espera
Inatividade por mais de 1 hora

Estados de espera no CTSS

- Existem diversas categorias de estados de espera no CTSS
- Os estados diferem entre si conforme o evento que esteja sendo aguardado. Essas categorias são as seguintes
  - esperando entrada – jobs que tentem ler a partir da região do Buffer de saída da console, mas com o buffer ainda vazio
  - esperando saída – jobs que tenham preenchido sua área de buffer de saída de console. Esses entram em um estado de espera até que a saída tenha sido encerrada
  - esperando arquivo – jobs aguardando para ler um arquivo compartilhado esteja então em estado bloqueado
  - aguardando comando – jobs que acabaram de efetuar a entrada de um comando e estão aguardando para serem classificados em algum nível de prioridade no estado "operando"
  - dormindo – jobs que foram interrompidos enquanto estavam recebendo algum serviço solicitado. O Job pode continuar a partir do ponto em que foi interrompido, ou então aceitar um novo comando
  - morto – jobs que tenham terminado os seus comandos anteriores, e que estão aguardando o próximo comando

- É preciso notar ausência de estados de espera de disco ou de fita bloqueado
- Os Jobs não são bloqueados enquanto estiverem fazendo entrada e saída em disco, ou seja, o processador continua alocado para ele enquanto estiver executando operações de entrada e saída em disco ou fita
- Convém ainda lembrar que existe um único Job completo na memória de cada vez, portanto, não é possível fazer overlap de entrada e saída convencional, como ocorre na multiprogramação


Usuário
Input wait
Output wait
File wait
Buffer de entrada da console vazio
Entrada recebida
Buffer de saída da console cheio
Final de impressão de saída
Arquivo bloqueado
Arquivo desbloqueado
Ligação telefônica
Comando terminado
Running
Interrupção
Limite de Quantum
Process scheduler 2
Submit
Interrupção
Ligação completada
Dormant
Restart
Working (ready)
Logout
Login
Entrada de comando
Atrib. prioridade
Comando logout
Dead
Entrada de comando
Waiting command
Terminate
Job scheduler 1
Estados de espera
Inatividade por mais de 1 hora

job scheduler

- A nuvem 1 da figura representa o job scheduler
- No CTSS o scheduler é muito simples:
- Há um número específico máximo de usuários (por exemplo 30) permitidos ao mesmo tempo
- Assim, os 30 primeiros clientes são aceitos em seu login
- Se já houver 30 usuários no sistema, outros não serão aceitos
- Se usuários adicionais tentarem conectar-se pelo telefone ao sistema, este interromperá a ligação telefônica
- Nenhuma informação é coletada ou mantida no sistema acerca dos usuários nessa situação

- Um job scheduler simples limita a carga do sistema com eficácia porém encoraja atitudes anti-sociais de diversos tipos
- Por exemplo, suponha que 10 alunos estejam envolvidos em um projeto
- Cada um deles combina com os demais que chegará bem cedo ao trabalho duas vezes por mês cada qual em dias diferentes do mês
- Quando o madrugador chega ao trabalho, ele liga 10 terminais, os quais serão mantidos conectados o dia inteiro, em benefício do grupo

Extensão do job scheduler

- O job scheduler foi estendido para lidar com essa situação
- Todos os usuários são associados em grupos
- A maioria dos usuários já pertencem a grupos de pesquisa no MIT
- Cada grupo recebe um número específico de linhas primárias
- Suponha que a esse grupo antes mencionado sejam associadas 2 linhas primárias
- Os dois primeiros usuários deste grupo a fazerem login recebem essas duas linhas primárias
- Caso o sistema esteja lotado nenhum usuário adicional desse grupo será aceito no login
- Por outro lado se o sistema não estiver lotado membros adicionais desse grupo serão aceitos no sistema, e a eles serão alocadas linhas secundárias, ou de standby
- Se um usuário de outro grupo com linha primária disponível deseja fazer login o sistema seleciona algum usuário de linha secundária reforça seu logout

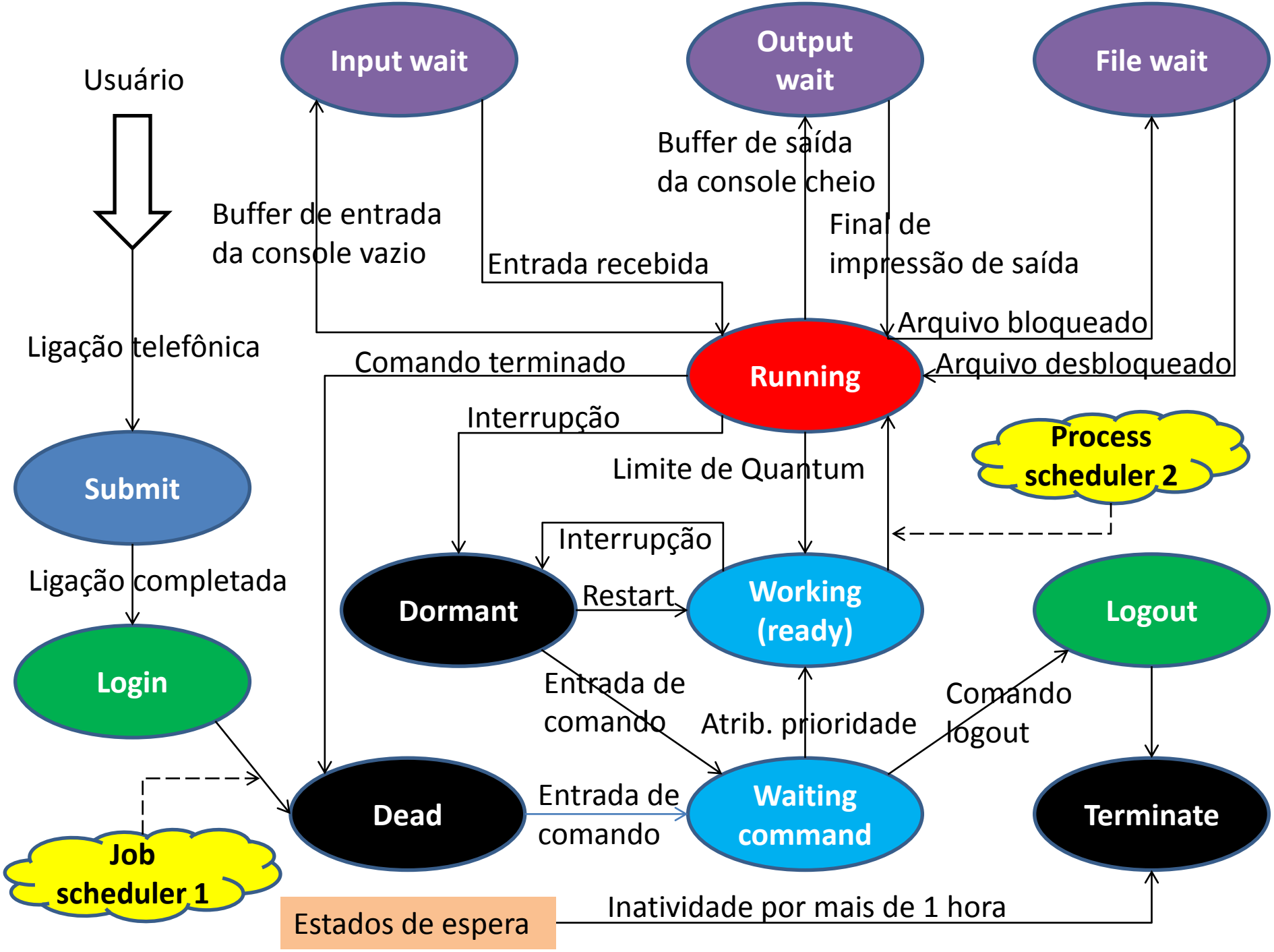
- O algoritmo de escolha do usuário a ser desconectado deve mudar periodicamente, para evitar que os usuários (que não desejam ser desconectados) monopolizem o sistema
- Uma das versões desse algoritmo desconecta aquele usuário que gastar o maior tempo de processador naquela ocasião
- A justificativa é de que aquele usuário já aproveitou a sua oportunidade
- Alguns espertinhos observam cuidadosamente seus próprios números e quando percebem ter usado mais processamento que a maioria dos colegas, fazem novo login zerando assim o seu contador de tempo
- Além de limitara carga a 30 usuários o job scheduler dá ao usuário que completar a ligação 2 minutos para fazer login, desconectando-o se ele não se identificar a tempo
- Um dos aspectos dos algoritmos de scheduling que se mostra mais atraente e também mais frustrante é a imprevisibilidade dos seus efeitos

Um algoritmo

- Um exemplo baseado no CTSS pode ilustrar este ponto: Como é que foi determinado o limite de 30 usuários?
- Assumiu-se que cada usuário impusesse uma certa carga ao sistema
- Mas o comportamento do usuário não é sempre o mesmo e esse limite se revelava ora muito restritivo hora muito permissivo
- Um Job scheduler especial foi projetado para monitorar a carga do sistema como um todo e caso esta carga venha a ultrapassar um certo limiar, ele automaticamente descontinua um dos jobs ativos, para reduzir a carga
- Vamos rastrear o comportamento desse esquema de balanceamento de carga na hipótese de que a carga atual esteja acima do ponto crítico
- Um procedimento de logout é acionado para remover esse job, ou seja, para fechar seus arquivos, atualizar informações contábeis, devolver ao sistema os recursos alocados etc.

Análise

- O procedimento de logout leva segundos para se completar
- Enquanto o logout transcorre, o algoritmo de balanceamento de carga verifica novamente a carga do sistema e constata que ela acabou aumentando
- Isso foi causado pela carga intensa causada pela execução do procedimento de logout
- Assim, em decorrência, um outro job ativo acaba sendo escolhido para ser descontinuado
- Isso aumenta ainda mais a carga e usuários adicionais vão sendo um a um removidos do sistema, e o fenômeno se repete até que algoritmo de balanceamento de carga seja finalmente satisfeito já que o número de jobs ativos é finito
- Não se sabe se esse algoritmo foi alguma vez utilizado, mas mesmo que seja hipotético, ele é um bom exemplo dos perigos potenciais que podem ser causados pela utilização de algoritmos “espertos” de scheduling.


Usuário
Input wait
Output wait
File wait
Buffer de entrada da console vazio
Entrada recebida
Buffer de saída da console cheio
Final de impressão de saída
Arquivo bloqueado
Arquivo desbloqueado
Ligação telefônica
Comando terminado
Running
Interrupção
Limite de Quantum
Process scheduler 2
Submit
Interrupção
Ligação completada
Dormant
Restart
Working (ready)
Logout
Login
Entrada de comando
Atrib. prioridade
Comando logout
Dead
Entrada de comando
Waiting command
Terminate
Job scheduler 1
Estados de espera
Inatividade por mais de 1 hora

Process scheduler

- A nuvem 2 da figura representa o process scheduler
- Diversos algoritmos foram experimentados, a maioria dos quais bastante sofisticados
- O objetivo do algoritmo do CTSS é dar uma resposta rápida ao usuário interativo, às custas do usuário não interativo que esteja abusando do uso do processador
- O algoritmo, ao atingir suas metas, estabelece 8 ready-queues (0 a 7)
- Os jobs de usuários interativos tendem a migrar para filas de números mais baixos enquanto os jobs CPU bound tendem a ocupar as filas de número mais alto

Dinâmica do scheduling

- O process scheduler executa em primeiro lugar Jobs que ocupam as filas de número baixo favorecendo assim os jobs interativos
- Jobs em filas de números mais altos recebem um quantum maior sempre que forem escalados para processar, reduzindo assim o overhead do tempo de Swapping para jobs grandes CPU bound
- Os jobs em cada fila recebem um quantum de tempo para sua execução (o intervalo de tempo desse quantum depende do número da fila que contém o job)

Quantum dinamicamente variável

- O número da fila é que determina a duração desse quantum. Sendo Q = 500 milissegundos o quantum básico, a duração do quantum (run quantum) para a fila n será:

  run quantum = $Q.2^n$ (n=número da fila)

- Inicialmente todos os jobs do usuário começam com n = 2 ou 3
- Se o job tiver tamanho menor que 4 Kb ele deverá começar no nível 2 (run quantum = 2 seg), e se tiver mais de 4Kb ele começará no nível 3 (run quantum = 4seg)

- Caso um Job não termine durante o quantum que lhe foi concedido, ele migra para o nível imediatamente mais alto
- Se o job corrente for executado por um tempo maior que o quantum do job que o interrompeu é permitido a ele efetuar preempção, ou seja, remover esse job e tomar o seu lugar
- Por exemplo, se um job de nível 7 estiver sendo executado, um job de nível 3 que se torne pronto não será removido da memória a não ser que o job de nível 7 tenha excedido pelo menos o tempo de processador já usado pelo job de nível 3
- Essa política tende a evitar a possibilidade de um job ser trazido para a memória e ser imediatamente desalojado por preempção
- O tempo de Swap do job é razoavelmente substancial no CTSS por causa da velocidade um tanto baixa do tambor utilizado para a realização do Swapping

- Há muitas outras regras que governam o tratamento da fila dos Jobs
- Quando o job do usuário recebe entradas pelo terminal ele é classificado como interativo e recebe nível 2 para scheduling
- Adicionalmente se um Job pronto para execução não for colocado no processador dentro do prazo de um minuto o Job é deslocado para a fila do nível imediatamente inferior

- Por exemplo considere-se um job que se desloca para baixo para a fila número 7
- Enquanto houver Jobs nas filas 0 a 6 ele jamais será executado
- O intervalo de um minuto assegura que se o Job não entrar em execução nesse intervalo ele será movido para o nível 6
- Esse limite garante que todo job deverá ser executado pelo menos uma vez em um intervalo de oito minutos mesmo que apenas por 500 milissegundos (caso ele tenha tido que atingir o nível 0 para ser executado)

- Para reduzir o overhead de examinar terminais não utilizados, ou seja, vagos, um job que permaneça no estado de inatividade por mais de uma hora é automaticamente terminado e removido do sistema
- Além dos jobs originados nos usuários online (foreground jobs) há muitos outros jobs no sistema
- Um job executa o FORTRAN Monitor System que opera como um sistema operacional em batch (background jobs)

- Esse job em background usualmente recebe um nível de prioridade mais baixo do que o do Job em foreground, uma vez que ele nunca recebe entrada de console sendo portanto tratado como um jovem não interativo
- Há muitos outros jobs de sistema chamados deamons, os quais examinam não continuamente o sistema de arquivos, fazendo cópias em fita dos arquivos recém criados ou recém modificados
- Isso garante a disponibilidade de uma cópia de backup relativamente atualizada no caso de ocorrer alguma falha grave no sistema

Administração de dispositivos

- Não havendo multiprogramação, a administração de dispositivos é trivial: só três tipos de operação de entrada e saída são executados
- 1. Console
  - Cada job recebe um buffer de entrada e outro de saída para entrada e saída em console
  - Toda entrada/saída em console é tratada pelo sistema operacional no banco A de memória
  - Entradas provenientes da consolesão depositadas na área de buffer do Job no banco de memória A, esteja ou não o Job correspondente presente no banco B de memória
  - Modificações apropriadas são efetuadas no estado do usuário quando os dados de entrada tiverem sido recebidos
  - Uma saída de console de um Job em execução é direcionada a sua área de buffer de saída no banco de memória A e, assim que possível, o sistema operacional executa a operação de entrada ou saída solicitada
  - Se a área de buffer estiver cheia, o job vai para um estado de out-wait

- 2. Disk I/O
  - Uma operação de entrada/saída em disco ou tambor acontece em duas situações: ou o usuário faz leitura em um arquivo ou então os procedimentos de administração de memória efetuam operações de Swap entre memória e disco
  - A leitura de arquivo por parte do usuário é executada sob demanda, e aquele usuário é, por essa razão, considerado como estando em processamento
  - Durante essa operação de entrada/saída o swapping é comandado pelo gerenciador de memória

- 3. Tape I/O -
  - O usuário inicia a entrada/saída em fita da mesma forma que o faz para entrada/saída em arquivo
  - Em outras palavras, esta operação exige que o programa esteja em execução nessa ocasião
  - A entrada/saída em impressora e em cartões são efetuados apenas por usuários FMS em background
  - Todas as entradas/saídas são direcionadas para uma fita, a qual é posteriormente levada fisicamente para um computador offline IBM 1401 para ser devidamente processada

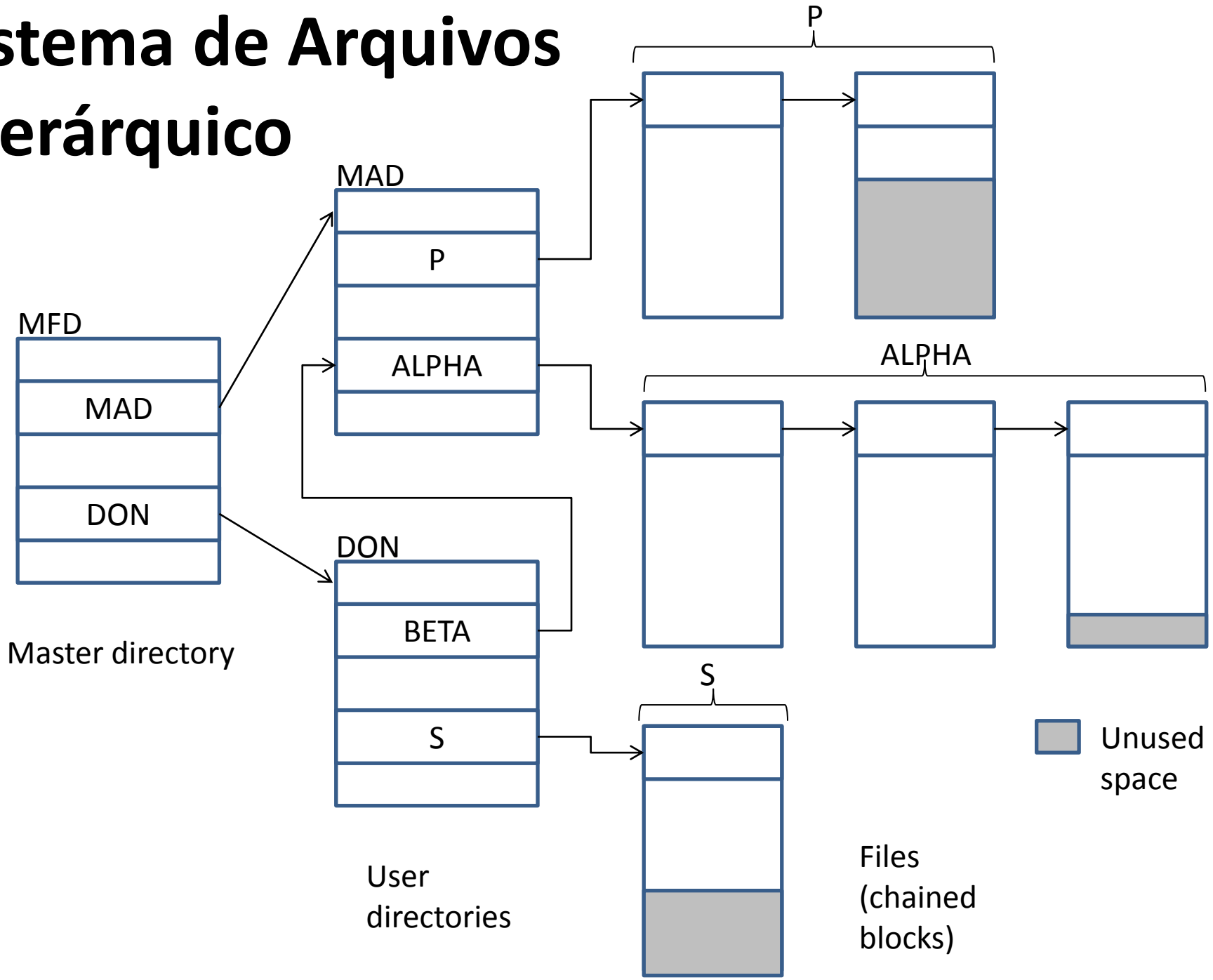
Administração de informação

- o sistema de arquivos do CTSS era na época e assim persistiu bastante como um dos sistemas mais avançados
- tinha propriedades e características que só viriam a ser imitadas muito mais tarde
- por exemplo nas mais anormais condições de logout (por exemplo na hora de desligar o sistema) o sistema salva o estado global de cada job forma que o usuário possa prosseguir sua execução em outra ocasião.

Características

- O sistema de administração de informações apresenta as seguintes características
  - O usuário pode escrever e manter programas e dados em disco
  - Programas de sistema são armazenados em disco e mantidos de forma similar aos programas e dados do usuário
  - Todos os arquivos são referenciados simbolicamente
  - Pode haver múltiplos arquivos simultaneamente acessíveis
  - O acesso a arquivos pode ser restringido (p. ex. read-only)
  - O formato padrão do registro físico é utilizado por todos esses arquivos, independente do formato dos registros lógicos (formato trivial, de um registro por trilha)
  - Um sistema de backup copia arquivos para fita automaticamente
  - É permitido o compartilhamento de arquivos entre usuários
  - Um projeto modular foi usado para simplificar a implementação e a manutenção

Sistema de Arquivos Hierárquico



Administração de informação

- A estrutura do arquivo é organizada em uma hierarquia de dois níveis conforme mostra a figura anterior
- Cada diretório de um usuário tem um elemento associado a cada arquivo
- Todos os registros físicos para um arquivo são encadeados entre si, como mostra a figura do slide anterior
- Isso torna eficiente o processamento sequencial (mais usual), porém os acessos aleatórios acabam se mostrando um tanto ineficientes