# Aula 1: introdução

Prof. DSc. Newton Spolaôr

Apoio: Prof. Sérgio Campos (UFMG), Marcelo Johann (UFRGS) e Daniel Abdala (UFU)

Disciplina Sistemas Operacionais
Bacharelado em Ciência da Computação
Universidade Estadual do Oeste do Paraná (UNIOESTE)

Brasil

19/08/2020

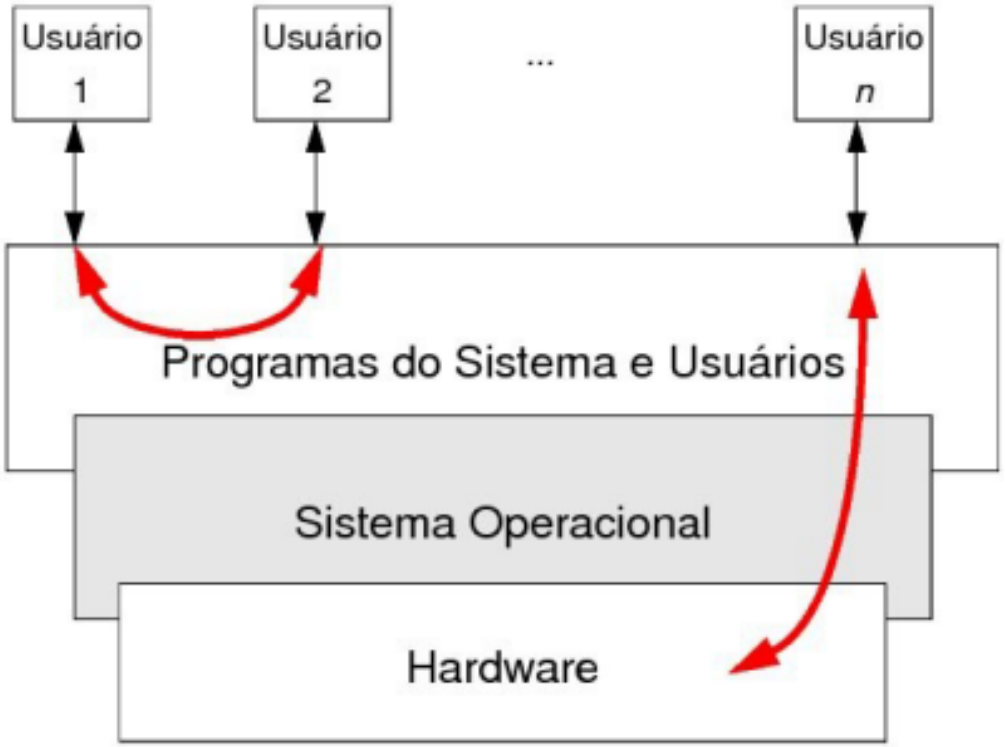
unioeste
Universidade Estadual do Oeste do Paraná

# Objetivo geral desta aula

Introduzir conceitos básicos de sistemas operacionais

# O que são sistemas operacionais? [1, 2]

- Um Sistema Operacional (SO) pode ser entendido como um software especial, o qual controla a interação entre programas, usuários e hardware



# O que são sistemas operacionais? [1, 2]

- Um SO também pode ser definido como um “ilusionista” que
  - Provê abstrações (modelos)
    - Algumas abstrações oferecem a usuários uma interface mais simples para acessar um hardware do que a própria interface do hardware

# O que são sistemas operacionais? [1, 2]

- Um SO também pode ser definido como um “ilusionista” que
  - Provê abstrações (modelos)
    - Algumas abstrações oferecem a usuários uma interface mais simples para acessar um hardware do que a própria interface do hardware
    - Outras abstrações possibilitam a execução de um mesmo programa em diferentes arquiteturas de computador

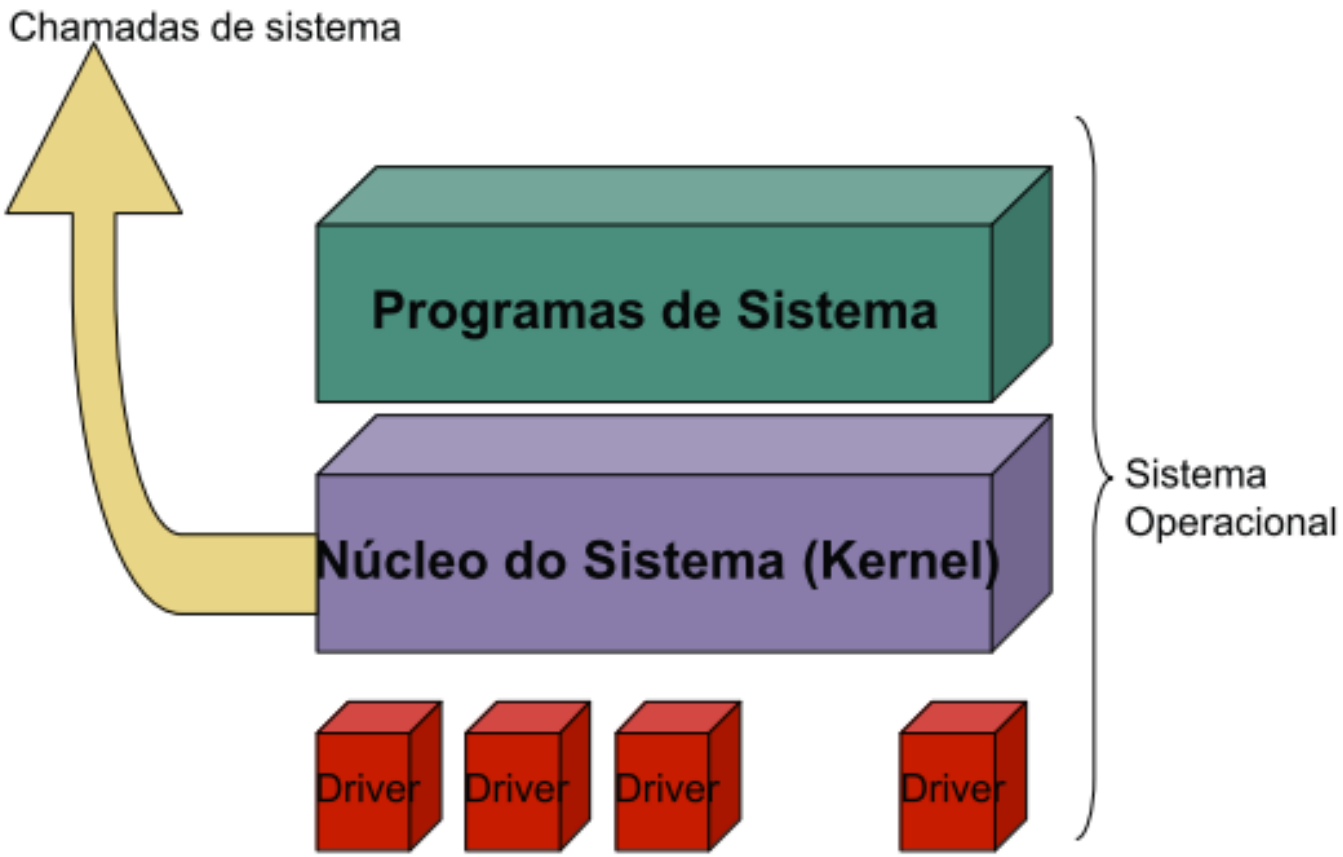
# O que são sistemas operacionais? [1]

- Um SO também pode ser definido como um “ilusionista” que
  - Coordena recursos
    - Visa otimizar o uso de recursos e permitir que um ou mais usuários ou programas realizem acesso paralelo a esses recursos

# O que são sistemas operacionais? [1]

- Um SO também pode ser definido como um “ilusionista” que
  - Coordena recursos
    - Visa otimizar o uso de recursos e permitir que um ou mais usuários ou programas realizem acesso paralelo a esses recursos
    - Exemplos de recursos incluem componentes de arquitetura, como processador, memória e dispositivos de entrada e saída

# O que são sistemas operacionais? [3]



# O que são sistemas operacionais? [3]

- Núcleo (*kernel*): conjunto mínimo de serviços executados pelo SO que define e escalona processos, entre outras tarefas
- Chamadas de sistema
  - Funções que os programas dos usuários podem usar para acessar os serviços do núcleo, como por exemplo ls, mkdir, cd, format e CTRL-C
  - O núcleo assume a execução dessas funções
- Programas de sistema: são serviços menos críticos, como compiladores, editores de texto, *shell*, GUI (Windows) e navegador (*browser*)

# Por que estudar sistemas operacionais? [1]

- O entendimento de SO permite compreender melhor computadores, incluindo o software que eles executam
- SO combinam conceitos de várias disciplinas da computação, tais como
  - Conceitos de linguagens de programação
  - Organização e arquitetura de computadores
  - Algoritmos e estruturas de dados

# Por que estudar sistemas operacionais? [1]

- O entendimento de SO permite compreender melhor computadores, incluindo o software que eles executam
- SO combinam conceitos de várias disciplinas da computação, tais como
  - Conceitos de linguagens de programação
  - Organização e arquitetura de computadores
  - Algoritmos e estruturas de dados
- SOs são importantes para a vida moderna...

# Por que estudar sistemas operacionais? [1]

# Por que estudar sistemas operacionais? [1]


globo esporte .com

campeonato mineiro

futebol | copa 2014 | f-1 | seu estado | na tv | + esportes | eu atleta | basquete

04/02/2013 15h53 - Atualizado em 04/02/2013 18h39

**Por problemas no Mineirão, Minas Arena recebe multa de R$ 1 milhão**

Governador de Minas considera graves as falhas na operação do estádio. Já ministro do Esporte afirma que 'não há justificativa' para os transtornos

**Ingressos**

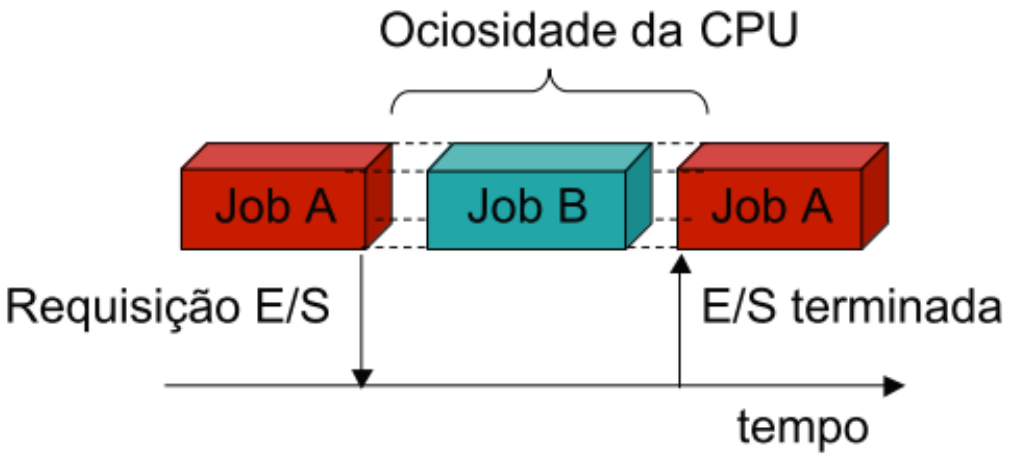
Houve lentidão no sistema. Enquanto um sistema processava, outro torcedor entrava e comprava, o que gerou a duplicidade de ingressos. Mas resolvemos isso, e alguns torcedores até foram acomodados em lugares melhores.

# Breve história de sistemas operacionais [1]

- Fase 0 (ausência de SO)
  - O usuário deveria estar presente todo o tempo
  - As tarefas de entrada e saída de dados, processamento e raciocínio do usuário ocorriam em sequência
  - Um problema da época consiste na ineficiência do computador, pois não havia sobreposição de tarefas

# Breve história de sistemas operacionais [4, 3, 1]

- Fase 1 (multiprogramação)
  - Sobreposição de tarefas, com vários programas carregados na memória ao mesmo tempo
  - Um dos objetivos era reduzir o desperdício de CPU devido às Entradas/Saídas




- Assim, mais de um *job* ou lote (grupo de programas com uso similar de recursos) estaria funcionando

# Breve história de sistemas operacionais [4, 3, 1]

- Fase 1 (multiprogramação)
  - Na verdade, somente um *job* está em execução no CPU por vez
  - Os demais *jobs* não executam, pois aguardam Entrada/Saída ou outro recurso

# Breve história de sistemas operacionais [4, 3, 1]

- Fase 1 (multiprogramação)
  - Na verdade, somente um *job* está em execução no CPU por vez
  - Os demais *jobs* não executam, pois aguardam Entrada/Saída ou outro recurso
  - Esses grupos de programas são armazenados em algum local, como disco
  - Em um determinado momento, há uma troca de *job* no CPU, de modo que um dos grupos de programas que aguardavam passa a executar na CPU, substituindo a antiga *job*

# Breve história de sistemas operacionais [4, 3, 1]

- Fase 2 (compartilhamento de tempo): a capacidade e o tempo de processamento do computador são divididos entre múltiplos usuários, cada um acessando seu próprio terminal – “multiprogramação para múltiplos usuários”

# Breve história de sistemas operacionais [4, 3, 1]

- Fase 2 (compartilhamento de tempo): a capacidade e o tempo de processamento do computador são divididos entre múltiplos usuários, cada um acessando seu próprio terminal – “multiprogramação para múltiplos usuários”
- Fase 3 (computadores pessoais): sistemas dedicados a um único usuário voltados para *desktops* e *notebooks*

# Breve história de sistemas operacionais [5, 4, 3, 1]

- Outros tipos de sistemas operacionais com importância histórica
  - SO distribuídos: recursos poderiam não estar no computador do usuário, mas em um outro computador na mesma rede, sendo acessados de modo transparente
  - SO de tempo real
    - Sistemas que especificam requisitos de tempo relativamente pequenos para realizar tarefas, podendo ser *hard real time* (garantia de cumprimento dos requisitos) ou *soft real time* (geralmente cumpre requisitos)
    - Alguns exemplos de propósito geral ou para sistemas embarcados: OSE, ChronOS e Apache Mynewt

# Sistemas operacionais modernos [1]

- Enormes, com centenas de milhares de linhas de código
- Complexos, pois...
  - podem gerenciar uma grande variedade de equipamentos e tecnologias
  - são sujeits a erros
  - exibem comportamento difícil de prever – SO podem executar por longos períodos de tempo

# Sistemas operacionais modernos [1]

- Exemplos: Windows, Linux, Mac OS, Android e iOS
- Pontos em comum entre esses sistemas incluem
  - Implementação de ideias de fases anteriores no histórico, como multiprogramação, compartilhamento de tempo e suporte a computadores pessoais (vide exemplos práticos)
  - Gerenciamento de recursos, como processador, memória, disco e dispositivos de entrada e saída

# Breve histórico do Windows [6, 7, 3]

- Windows NT 3.1 (1993): multi-programação com único usuário para computadores com CPU de 32 bits, sendo uma “casca gráfica” para o MS-DOS

# Breve histórico do Windows [6, 7, 3]

- Windows NT 3.1 (1993): multi-programação com único usuário para computadores com CPU de 32 bits, sendo uma “casca gráfica” para o MS-DOS
- Windows NT 4.0 (1995): interface gráfica aperfeiçoada

# Breve histórico do Windows [6, 7, 3]

- Windows NT 3.1 (1993): multi-programação com único usuário para computadores com CPU de 32 bits, sendo uma “casca gráfica” para o MS-DOS
- Windows NT 4.0 (1995): interface gráfica aperfeiçoada
- Windows 2000 (1999): variações para cliente e servidor, sendo escrito conforme paradigma orientado a objetos e suportando múltiplos usuários e esquema NTFS

# Breve histórico do Windows [6, 7, 3]

- Windows NT 3.1 (1993): multi-programação com único usuário para computadores com CPU de 32 bits, sendo uma “casca gráfica” para o MS-DOS
- Windows NT 4.0 (1995): interface gráfica aperfeiçoada
- Windows 2000 (1999): variações para cliente e servidor, sendo escrito conforme paradigma orientado a objetos e suportando múltiplos usuários e esquema NTFS
- Windows XP (2001): melhor segurança via *firewalls* e suporte a computadores com CPU de 64 bits

# Breve histórico do Windows [6, 7, 3]

- Windows NT 3.1 (1993): multi-programação com único usuário para computadores com CPU de 32 bits, sendo uma “casca gráfica” para o MS-DOS
- Windows NT 4.0 (1995): interface gráfica aperfeiçoada
- Windows 2000 (1999): variações para cliente e servidor, sendo escrito conforme paradigma orientado a objetos e suportando múltiplos usuários e esquema NTFS
- Windows XP (2001): melhor segurança via *firewalls* e suporte a computadores com CPU de 64 bits
- Windows mais recentes: melhorias em funcionalidades para multimídia, redes, segurança e interface gráfica

# Breve histórico do Linux [8, 3]

- Linux *kernel* 0.01 (1991): sistema para computadores com CPU como Intel 386

# Breve histórico do Linux [8, 3]

- Linux *kernel* 0.01 (1991): sistema para computadores com CPU como Intel 386
- Linux *kernel* 1.0 (1994): suporte a rede

# Breve histórico do Linux [8, 3]

- Linux *kernel* 0.01 (1991): sistema para computadores com CPU como Intel 386
- Linux *kernel* 1.0 (1994): suporte a rede
- Linux *kernel* 2.0 (1996): suporte a computadores com CPU de arquiteturas distintas, como SPARC, e implementação de programação paralela (*threads*) no *kernel* (núcleo)

# Breve histórico do Linux [8, 3]

- Linux *kernel* 0.01 (1991): sistema para computadores com CPU como Intel 386
- Linux *kernel* 1.0 (1994): suporte a rede
- Linux *kernel* 2.0 (1996): suporte a computadores com CPU de arquiteturas distintas, como SPARC, e implementação de programação paralela (*threads*) no *kernel* (núcleo)
- Linux *kernel* 2.4 (2002): melhoria no escalonamento (agendamento para execução) de *threads*

# Breve histórico do Linux [8, 3]

- Linux *kernel* 0.01 (1991): sistema para computadores com CPU como Intel 386
- Linux *kernel* 1.0 (1994): suporte a rede
- Linux *kernel* 2.0 (1996): suporte a computadores com CPU de arquiteturas distintas, como SPARC, e implementação de programação paralela (*threads*) no *kernel* (núcleo)
- Linux *kernel* 2.4 (2002): melhoria no escalonamento (agendamento para execução) de *threads*
- Linux mais recentes: melhorias em funcionalidades para interface gráfica, redes, entrada/saída e variações para dispositivos variados (Android e ChromeOS) □

# Alguns serviços de sistemas operacionais [10]

- *Boot* (iniciação)
- Gerência de processador (e processos)
- Gerência de memória
- Gerência de sistema de arquivos
- Gerência de entrada e saída

# *Boot* [10]

- O *boot* de um sistema pode ser dividido nos seguintes passos
  1. Carrega e executa o programa de iniciação localizado em memória ROM ou FLASH
  2. Executa o *Power-on Self-Test* (POST)
  3. Busca controladores de dispositivos para o SO
  4. Carrega o SO de algum dispositivo de armazenamento secundário para a memória principal (RAM)

# *Boot* [10]

- Passos do POST
  1. Garantir que pelo menos um dispositivo padrão de Entrada e um de Saída estejam presentes
  2. Testa se a memória primária está presente, quanta memória está disponível e se ela está funcionando corretamente
  3. Executa pequenos programas que “verificam” se tais dispositivos estão presentes e se estão funcionando corretamente

# *Boot* [10]

- Passos do POST
    4. Testa o dispositivo secundário de armazenamento (HD)
    5. Passa controle ao sistema de *bootstraping* para carregar o Sistema operacional
- Tais programas estão localizados no *firmware Basic Input/Output System* (BIOS)

# Gerência de processador (e processos) [10]

- *Boot* (iniciação)
- Parte do SO que cuida da alocação do processador – CPU
- Em sistemas multitarefa, esta é uma das partes críticas do SO, pois a alocação eficiente da CPU(s) impacta diretamente no desempenho do sistema
- Processo: um programa “apto a ser executado”

# Gerência de processador (e processos) [10]

- Tópicos relacionados
  - Representação de processos (estruturas de dados)
  - Escalonamento (agendamento) de processos
  - Comunicação entre processos

# Gerência de memória [10]

- Responsável por alocar memória para processos
- Como muitos processos executam concorrentemente, a quantidade de memória real é muito menor que a memória virtual total utilizada por todos os processos
- O gerenciador de memória é responsável por criar essa “ilusão” de uma memória muito maior do que realmente existe

# Gerência de memória [10]

- Tópicos relacionados
  - Conceito do espaço de endereçamento
  - Memória virtual
  - Páginas e segmentos de memória

# Gerência de sistema de arquivos [10]

- Responsável pela gerenciamento de todos os arquivos do SO
- Implementa e controla as abstrações referentes a arquivos e gerencia o acesso a elas
- Arquivo: conceito fundamental em virtualmente todos os SO modernos
- Abstrai dados armazenados em diversos tipos de dispositivos, provendo uma única interface comum

# Gerência de sistema de arquivos [10]

- Sistema de Arquivos: forma de organizar a informação contida nos dispositivos de armazenamento
- Conceitos associados: diretório, *link* e pseudoarquivos

# Gerência de entrada e saída [10]

- Parte do SO responsável por controlar a Entrada e Saída (E/S) de dados do sistema computacional;
- Uma parte da gerência de E/S é comum a todos os dispositivos de E/S
- Devido à complexidade e variedade de dispositivos de E/S, uma parcela deles necessita de tratamento especializado
  - Vídeo
  - Rede

# Estrutura de sistemas operacionais [10]

- Como SO é um programa, a estrutura do SO é, basicamente, a estrutura de um programa
- Existem diferentes maneiras de estruturar SO, tais como: monolítico, em camadas e micronúcleo ($\mu$*kernel*)

# SO monolítico [10]

- Organiza SO em um único programa
  - Estruturação em diversar rotinas
  - Interface entre sistemas funcionais é difusa
- Alguns SO monolíticos permitem o carregamento *a posteriori* de *drivers* de dispositivos
- Consiste em uma das principais estruturas utilizadas atualmente, como ilustrado por Linux e Windows

# SO em camadas [10]

- Isolamento de funcionalidades e divisão de responsabilidades
  - Maior segurança, coesão e acoplamento
  - Menor desempenho
- Exemplo: MULTICS

# SO de micronúcleo ($\mu$*kernel*) [10]

- Núcleo reduzido, com poucos processos, enquanto vários serviços são oferecidos por processos a nível de usuário (fora do *kernel*)
- Ideia básica: alcançar alta confiabilidade por meio de subdivisão em módulos pequenos
- Muito utilizado em pesquisa e desenvolvimento de novos SO
- Exemplo: Symbian e MINIX3

# SO cliente-servidor [10]

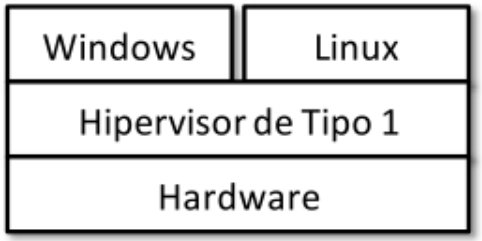
- Distinção entre duas classes de processos
  - Processos servidores: prestam serviços
  - Processos clientes: utilizam serviços
- Comunicação realizada via troca de mensagens
- Cliente/Servidor pode ser executado na mesma máquina ou em máquinas distintas

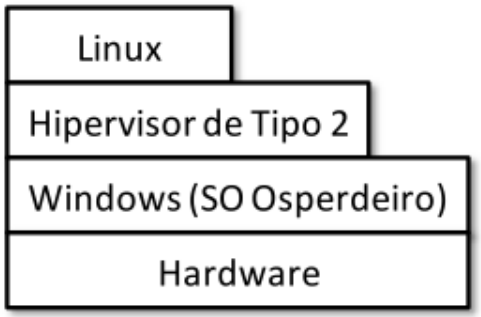
# SO no formato de máquina virtual [10]

- Habilitam um computador a rodar multiplos SO ao mesmo tempo
- Usos
  - Gerenciar aplicações legadas
  - Otimizar utilização do hardware
  - Portabilidade
- *Virtual Machine Monitor* (VMM), ou Hipervisor: camada de software mais básica que apoia o funcionamento de máquinas virtuais
- Exemplos incluem VMWare e Xenserver

# SO no formato de máquina virtual [10, 11]

- Exemplos de estruturação de hipervisores



# SO no formato de máquina virtual [10, 11]

- A Máquina Virtual Java (MVJ) é outro exemplo famoso
  - Criada junto com a linguagem Java

# SO no formato de máquina virtual [10, 11]

- A Máquina Virtual Java (MVJ) é outro exemplo famoso
  - Criada junto com a linguagem Java
  - O compilador Java produz código para MVJ, o qual é então tipicamente executado por um interpretador MVJ

# SO no formato de máquina virtual [10, 11]

- A Máquina Virtual Java (MVJ) é outro exemplo famoso
  - Criada junto com a linguagem Java
  - O compilador Java produz código para MVJ, o qual é então tipicamente executado por um interpretador MVJ
  - Uma vantagem é que o código para MVJ pode ser transferido pela Internet para qualquer computador que tenha interpretador MVJ e executar lá

# SO no formato de máquina virtual [10, 11]

- A Máquina Virtual Java (MVJ) é outro exemplo famoso
  - Criada junto com a linguagem Java
  - O compilador Java produz código para MVJ, o qual é então tipicamente executado por um interpretador MVJ
  - Uma vantagem é que o código para MVJ pode ser transferido pela Internet para qualquer computador que tenha interpretador MVJ e executar lá
  - Se, por outro lado, o compilador tivesse produzido binário para Pentium, seria mais complexo transferi-lo e interpretá-lo posteriormente em um computador SPARC

# Considerações finais

# Considerações finais

- Nesta aula foram apresentados conceitos básicos de sistemas operacionais

# Considerações finais

- Nesta aula foram apresentados conceitos básicos de sistemas operacionais
- Na próxima aula, será tratado o tema de processos

# Contato

newton.spolaor@unioeste.br

# Referências bibliográficas I

[1] Sérgio Campos and Marcus Rocha. Sistemas operacionais. http://homepages.dcc.ufmg.br/ scampos/cursos/so/, 2002. Notas didáticas.

[2] A. Silberschatz and P. Galvin. *Operating systems concepts*. Addison-Wesley, 5a edition, 1997.

[3] Marcelo Johann. Sistemas operacionais, 2009. Notas didáticas.

[4] Roberta Lima Gomes. Sistemas operacionais. http://www.inf.ufes.br/ rgomes/so.htm, 2016. Notas didáticas.

# Referências bibliográficas II

[5] Wikipedia. Real-time operating system. https://en.wikipedia.org/wiki/Real-time_operating_system, 2018.

[6] Wikipedia. Microsoft windows. https://en.wikipedia.org/wiki/Microsoft_Windows, 2017.

[7] Igor Augusto de Carvalho Alves. Sistemas operacionais, 2012. Notas didáticas.

[8] Wikipedia. Linux. https://en.wikipedia.org/wiki/Linux, 2017.

# Referências bibliográficas III

[9] Globo. Android passa windows e se torna o sistema operacional mais usado do mundo. http://g1.globo.com/tecnologia/noticia/android-passa-windows-e-se-torna-o-sistema-operacional-mais-usado-do-mundo.ghtml, 2017.

[10] Daniel Abdala. Sistemas operacionais, 2016. Notas didáticas.

[11] A. S. Tanenbaum. *Modern Operating Systems*. Pearson education, 3rd edition, 2009.