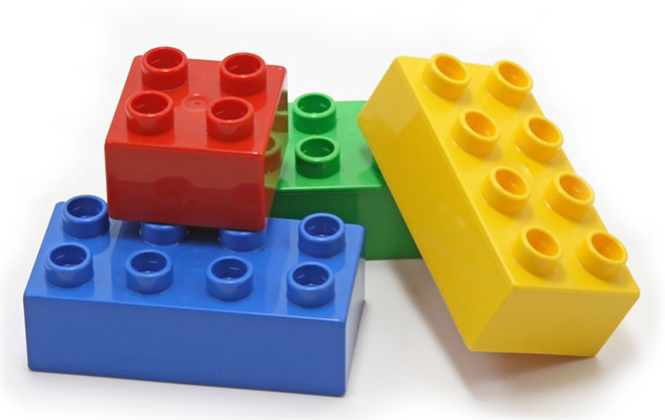
Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



4. Desenvolvimento no AOSP

Yuri Assayag
yfa@icomp.ufam.edu.br

ColabWeb: bit.ly/devtitans-moodle

Disciplinas

- Linux
  - Fundamentos do Linux
  - Comandos, Uso do Terminal, Interpretador de Comandos
  - Sistema de Arquivos, Processos, Segurança
- Desenvolvimento no AOSP
  - Baixando e Compilando
  - Explorando o Código Fonte do AOSP
  - Explorando o Android/Linux
  - Sistema de Inicialização do Android
  - Criando Novo Produto
  - Personalização do Produto
  - Overlay, Módulos, Bibliotecas, Apps

Metodologia

- Aulas Teóricas
  - Visão geral do conteúdo
  - 30 a 60 minutos de teoria
  - Rápidos exemplos práticos

- Laboratórios Práticos
  - Estilo tutorial
  - Complementam o conteúdo teórico
  - Laboratórios duram entre 40min e 1h
    - *Se não der tempo de terminar, não tem problema*
    - *Pode ser feito após o término da aula ou na próxima semana*
  - Professor e monitor disponíveis para dúvidas

Avaliação

- Laboratórios Práticos (LP)
- Presença
- Pontos extras por participação e desafios

Laboratórios Práticos

- Todos os laboratórios são **individuais**
  - Pode discutir e tirar dúvidas entre si

- Necessidade de um PC de **alta performance** com **Linux**
  - O AOSP só roda/compila no Linux
  - Configurações recomendadas:
    - *Processador: i7*
    - *Memória: 16GB*
    - *Disco: 400GB (de preferência, SSD)*

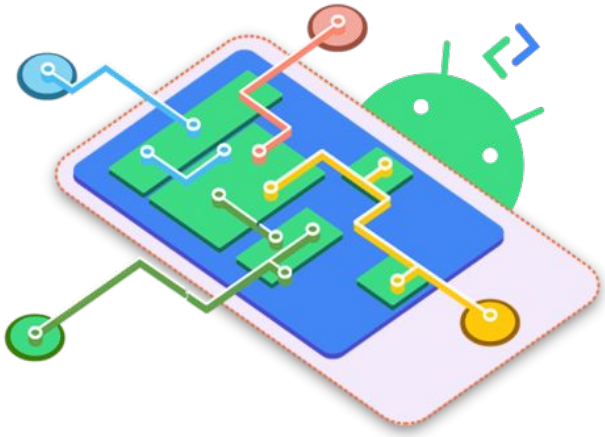
Dúvidas

- Presencial
  - Marcar com o professor ou monitor na UFAM.

- Dúvidas poderão ser tiradas também por **E-Mail**:
  - Professor: yfa@icomp.ufam.edu.br
  - Monitor: mop@icomp.ufam.edu.br

- Videoconferência ou Telegram
  - Qualquer aluno pode solicitar um atendimento
  - Dia e horário a combinar

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



Android Open Source Project

ColabWeb: bit.ly/devtitans-moodle

Yuri Assayag
yfa@icomp.ufam.edu.br

Android

- Sistema operacional para dispositivos embarcados móveis mais usado no mundo
  - Instalado em aproximadamente 80% dos smartphones
- Código-fonte aberto
  - Pode ser adaptado e instalado em diversos dispositivos de fabricantes diferentes
  - Pode ser usado com propósitos diferentes que vão muito além dos smartphones
    - *eletrodomésticos, veículos e relógios*
- Essa diversidade resultou em
  - Contribuições ao projeto vindas dos mais diversos lugares
  - Convergindo em um dos projetos de software mais inovadores do mundo

Android Open Source Project – AOSP

- Ponto de partida para a personalização e criação de um novo sistema baseado em Android
- O projeto contém:
  - Informações e documentações
  - Repositório de código-fonte
  - Testes de validação e de compatibilidade
- Principal responsável por garantir a compatibilidade entre dispositivos
  - Todos os fabricantes se baseiam no mesmo código-fonte

Android Open Source Project – AOSP

- O AOSP não só distribui o código-fonte; aceita também contribuições
  - Tais implementações poderão ser, eventualmente, incorporadas à solução final
  - Tal característica faz com que o Android seja um dos maiores projetos cooperativos da atualidade

- Você está prestes a embarcar em uma jornada que o fará parte desse ecossistema de profissionais que mantêm e aprimoram o Android!

Repositório

- O Android/AOSP é baseado em diversos projetos de software livre
  - Mantidos por desenvolvedores diferentes
  - Alguns do próprio AOSP
  - Outros de projetos diferentes:
    - *Linux Kernel, SQLite, FreeType, LibPng, LibXml, ZLib*

- Cada projeto individual é gerenciado usando o **Git**

Git

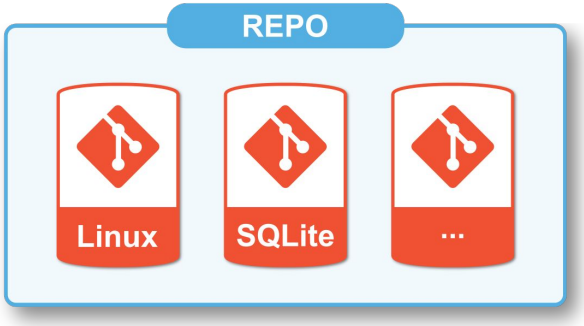
- Sistema distribuído de controle de versão

- Gerencia e rastreia mudanças em arquivos de projetos colaborativos
  - Normalmente, códigos-fonte

- Criado por Linus Torvalds em 2005
  - Para o desenvolvimento do kernel do Linux
  - Substituiu um sistema comercial/fechado usado antes

Repo

- Para gerenciar os vários repositórios Git, a ferramenta **Repo** é usada



- Repo:
  - Criado pelo Google
  - Unifica e gerencia diversos repositórios Git
  - Não substitui o Git, apenas facilita o trabalho com vários repositórios
    - *Na maioria dos casos, você continua usando o Git dentro dos subprojetos*

Instalando o Repo

```
$ mkdir bin; cd bin
$ wget https://storage.googleapis.com/git-repo-downloads/repo
$ chmod 755 repo
$ echo "export PATH=$PATH:~/bin/" >> ~/.bashrc
$ repo version
```

Inicializando o Repo do AOSP

```
$ cd ~; mkdir aosp; cd aosp

$ repo init --depth=1  \
    -u https://android.googlesource.com/platform/manifest  \
    -b android-13.0.0_r24
```

Baixando o AOSP

```
$ nproc --all
$ repo sync -c -j8
```

- Pressione Ctrl+C para cancelar o download e copie do disco local

```
$ cd ~
$ rm -rf aosp
$ mv aosp-original aosp
$ cd aosp
```

Explorando o Diretório do AOSP

```
$ ls
```

Diretório	Descrição
art	ART (*Android Runtime*), o ambiente de execução (máquina virtual Java) usado atualmente pelo Android.
bionic	O Bionic é a biblioteca C padrão do Android. Faz o papel do glibc (*GNU C Library*).
dalvik	O Dalvik é o antigo ambiente de execução (máquina virtual Java) do Android.
device	Armazena os produtos, personalizações do Android. Será bastante usado nos próximos laboratórios.
external	Projetos externos ao AOSP como o libssl (criptografia), webkit (biblioteca HTML e JavaScript), etc.
frameworks	Componentes da camada Framework, que provê serviços para os aplicativos Android (apps).
hardware	Camada HAL (*Hardware Abstraction Layer*). Provê acesso a componentes de hardware (audio, camera, etc).
kernel	Kernel do Linux e módulos já compilados. Contém também arquivos de configuração.
libcore	Biblioteca Java padrão (OpenJDK). Provê classes como ArrayList, String, Object, etc.
packages	Aplicações Android. Contém o código de apps como Settings, Calendar, Dialer, Launcher3, etc.
system	Bibliotecas e processos nativos do Android (liblog, vold, lmkd, e outros).

Explorando o AOSP

```
$ find libcore | grep String.java
```

Explorando o Código do AOSP

- O código-fonte do AOSP é a **principal** fonte atualizada de **documentação** do sistema
  - Documentações e livros da Internet estão todos desatualizadas
  - O AOSP evolui mais rapidamente que a documentação disponível
  - Códigos e comandos com mais de 1 ou 2 anos não funcionam mais

- Para facilitar a exploração do código-fonte do AOSP, é necessário um serviço feito para isso:
  - Android Code Search
    - *cs.android.com/android/platform/superproject/*

Android Code Search

- Desenvolvido pela Google para facilitar a exploração de códigos-fonte de projetos

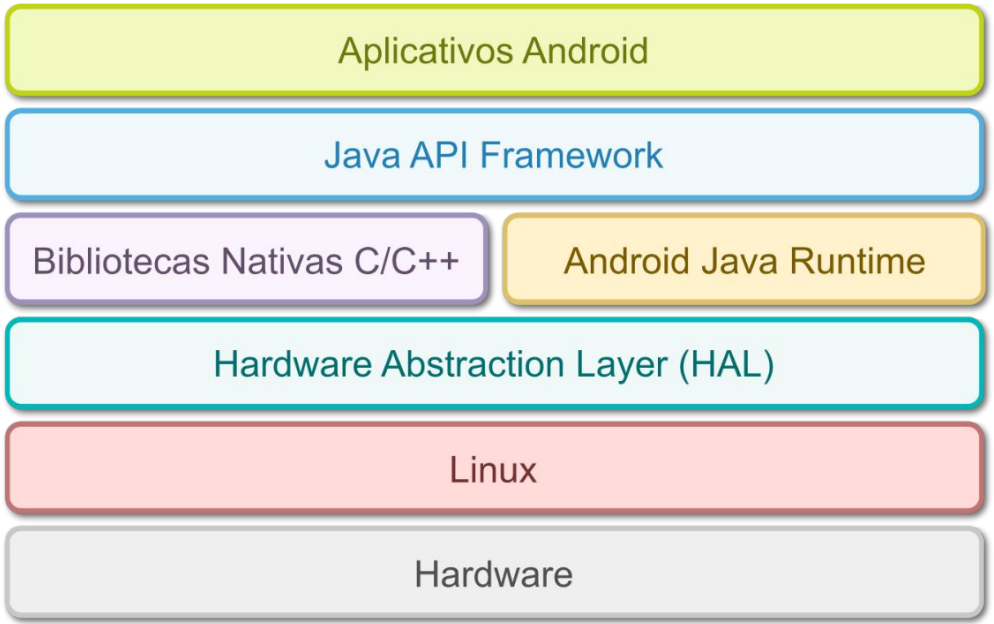
- Será a maior fonte de informações durante todo o curso
  - Será a maior fonte de informações durante todo o curso
  - É importantíssimo que você se acostume a usá-lo como fonte principal de informações sobre o AOSP
  - Faça um bookmark dele:

cs.android.com/android/platform/superproject/

Android Code Search

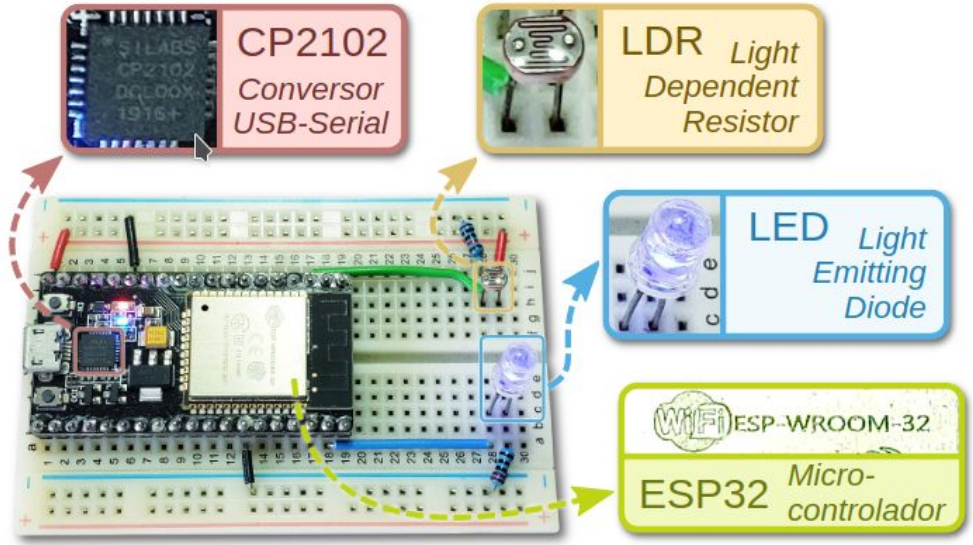
- Exemplo 1:
  - Busque por SurfaceFlinger.cpp
  - Encontre o método captureDisplay
- Exemplo 2:
  - Busque por packages/apps/Settings/res/values-pt/strings.xml

Camadas do Android

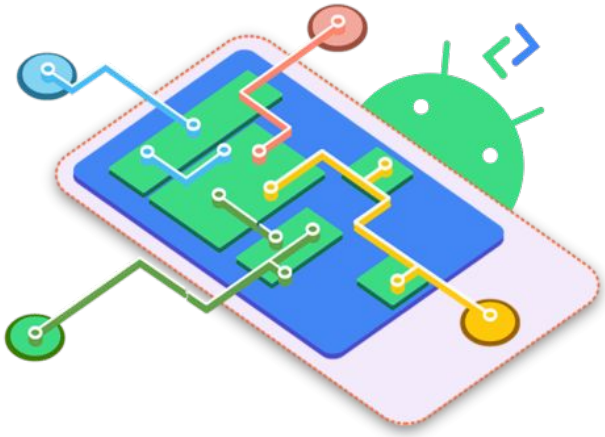


- O Android é dividido em Camadas

Dispositivo SmartLamp



Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



Compilando e Executando o AOSP

Yuri Assayag
yfa@icomp.ufam.edu.br

ColabWeb: bit.ly/devtitans-moodle

Sistema de Compilação do AOSP

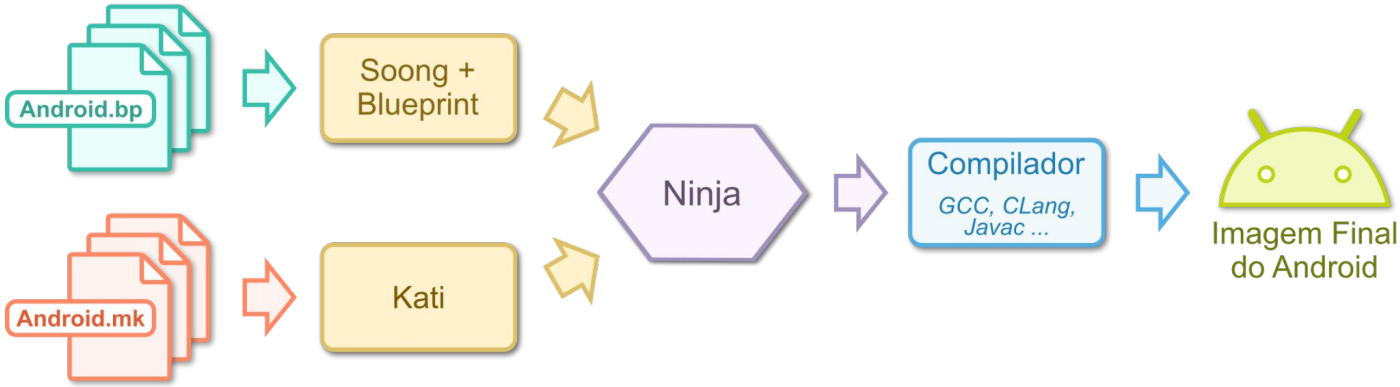
- Antigamente, o AOSP usava o **`make`**
  - Arquivos com o nome **`Android.mk`** eram usados para configurar a compilação
  - Entretanto, o make se tornou lento, devido ao tamanho do AOSP

- O **`make`** está sendo substituído pelo **Soong**
  - Usa arquivos de blueprint chamados: **`Android.bp`**

Sistema de Compilação do AOSP

- Os arquivos `Android.mk` ainda existentes são interpretados usando a ferramenta Kati
- Tanto o Soong quanto o Kati geram como resultado uma lista de arquivos que precisam ser compilados/processados
  - Esta lista é passada para a ferramenta Ninja
- A ferramenta Ninja irá realmente executar os comandos para compilar/processar os arquivos
  - Gcc, CLang, Javac …

Sistema de Compilação do AOSP



Configurando o Ambiente de Compilação

- Instalação dos pacotes necessários

```
$ sudo apt install openjdk-8-jdk build-essential ...
```

- Configuração do terminal

```
$ cd ~/aosp
$ source build/envsetup.sh
$ lunch
```

Configurando o Ambiente de Compilação

- O comando lunch configura o produto a ser compilado

```
$ lunch sdk_phone_x86_64-userdebug
```

- Veja as variáveis de ambiente criadas:
  - Tais variáveis serão usadas pelo sistema de compilação

```
$ export | grep "ANDROID_\|TARGET_\|PATH"
```

Compilando!

- Para compilar o Android, execute:

```
$ m
```

- Se for a primeira vez, a compilação pode demorar de 3 a 4 horas em uma máquina com as configurações recomendadas:
  - Processador: i7
  - Disco: SSD (>500GB)
  - Memória: 16GB
- Na segunda vez em diante, deve demorar entre 1 e 15 minutos, dependendo das mudanças feitas

Emulator

- Uso do Emulator para testar a compilação
  - Permite executar o Android no seu próprio computador emulando um celular
  - É o mesmo programa usado pelo Android Studio para executar aplicativos
- O Android é executado em uma máquina virtual chamada Android Virtual Device (AVD)
  - A máquina virtual usada pelo emulator é conhecida como **goldfish**
- Atualmente existe um novo emulador no AOSP chamado **cuttlefish**
  - Entretanto, continuaremos usando o emulator/goldfish por este ser mais estável, conhecido e ter recursos mais avançados como suporte a USB

Executando o Android no Emulator

- Para executar o emulator já abrindo o Android compilado, execute:

```
$ emulator
```

Modificando o Código-Fonte do Android

- Modificando string “build number” no arquivo:

```
$ nano packages/apps/Settings/res/values/strings.xml
```

- Recompilando

```
$ m
```

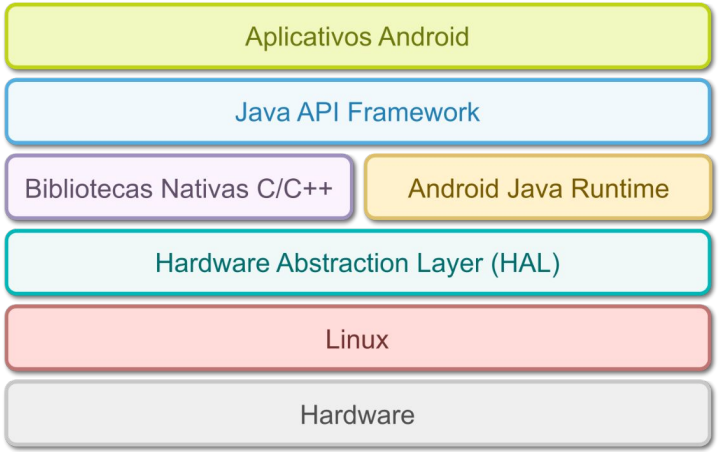
- Testando

```
$ emulator
```

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.1: Baixando o Código-Fonte do AOSP
  - Laboratório 4.2: Compilando, Executando e Modificando o Android

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados

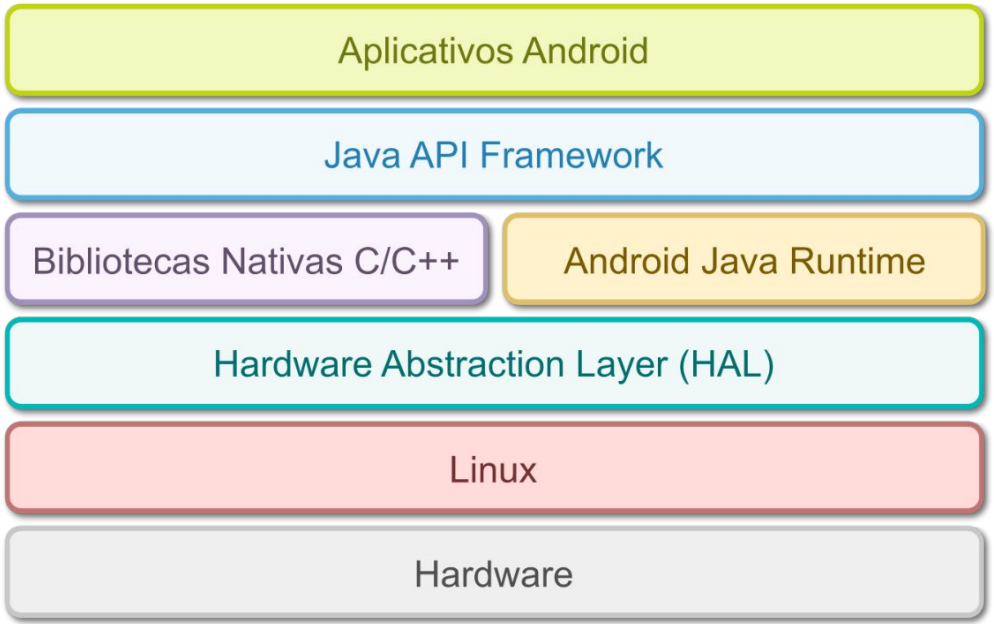


Explorando e Conhecendo o Android

Yuri Assayag
yfa@icomp.ufam.edu.br

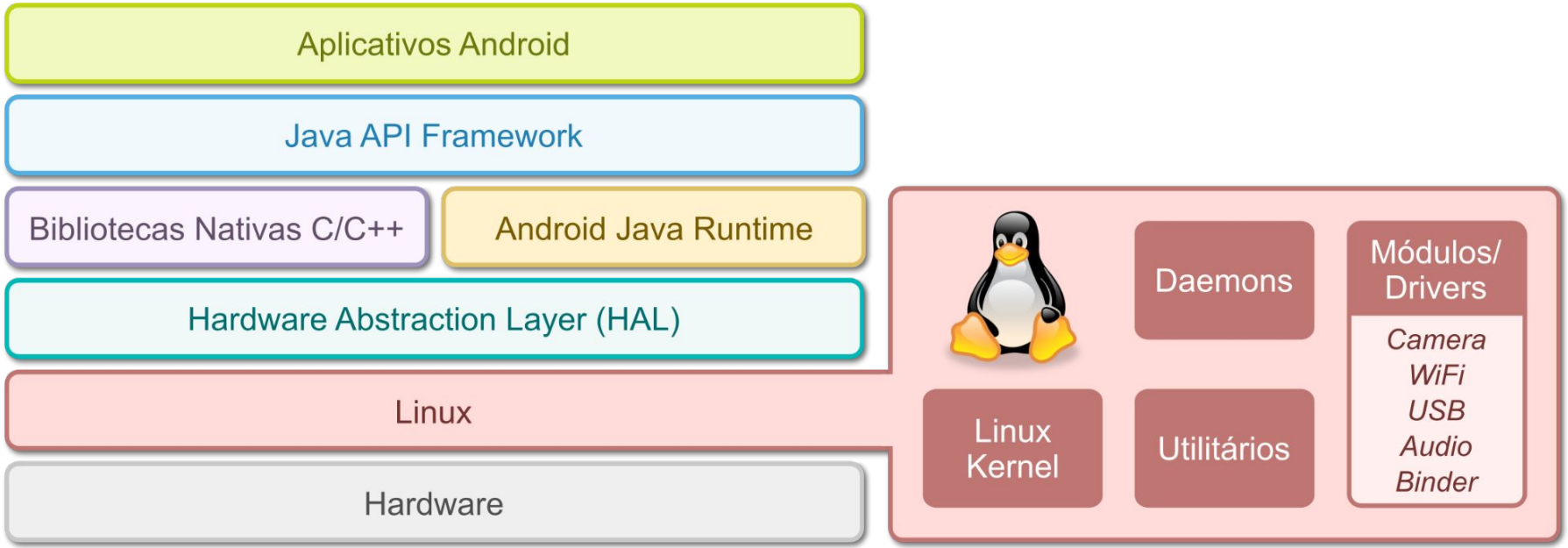
ColabWeb: bit.ly/devtitans-moodle

Camadas do Android

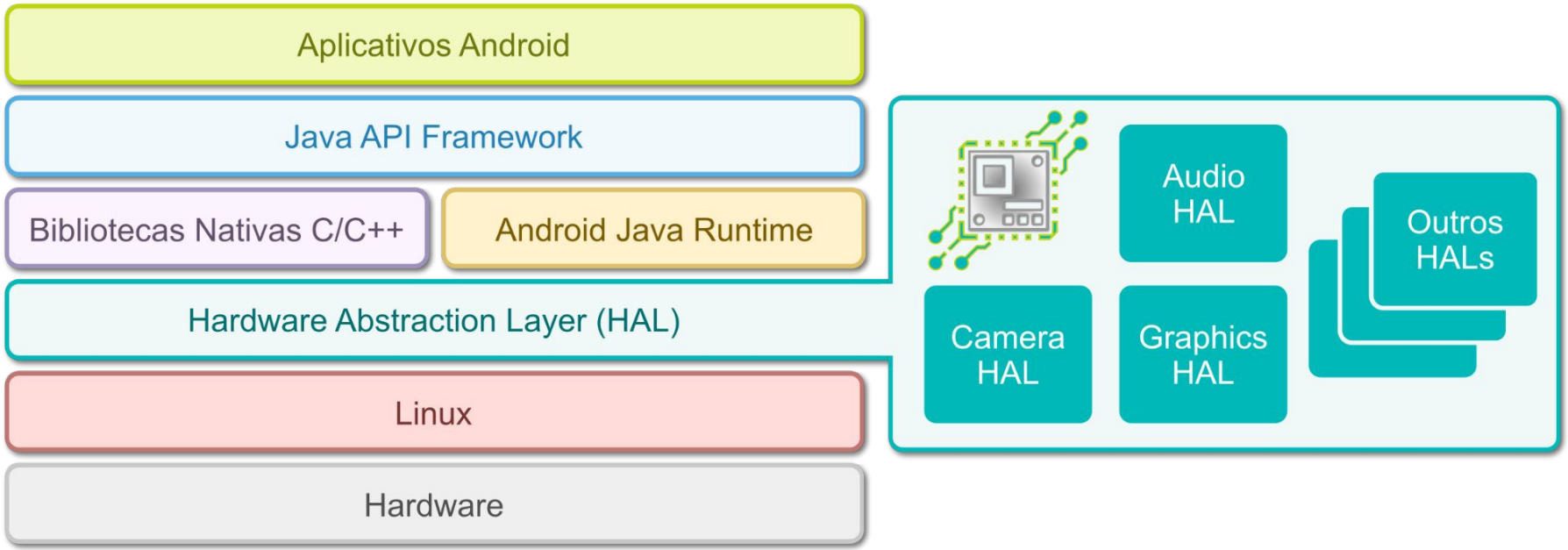


- O Android é dividido em Camadas
  - A primeira delas, de mais baixo nível, é o *hardware*
    - *Parte física do dispositivo*

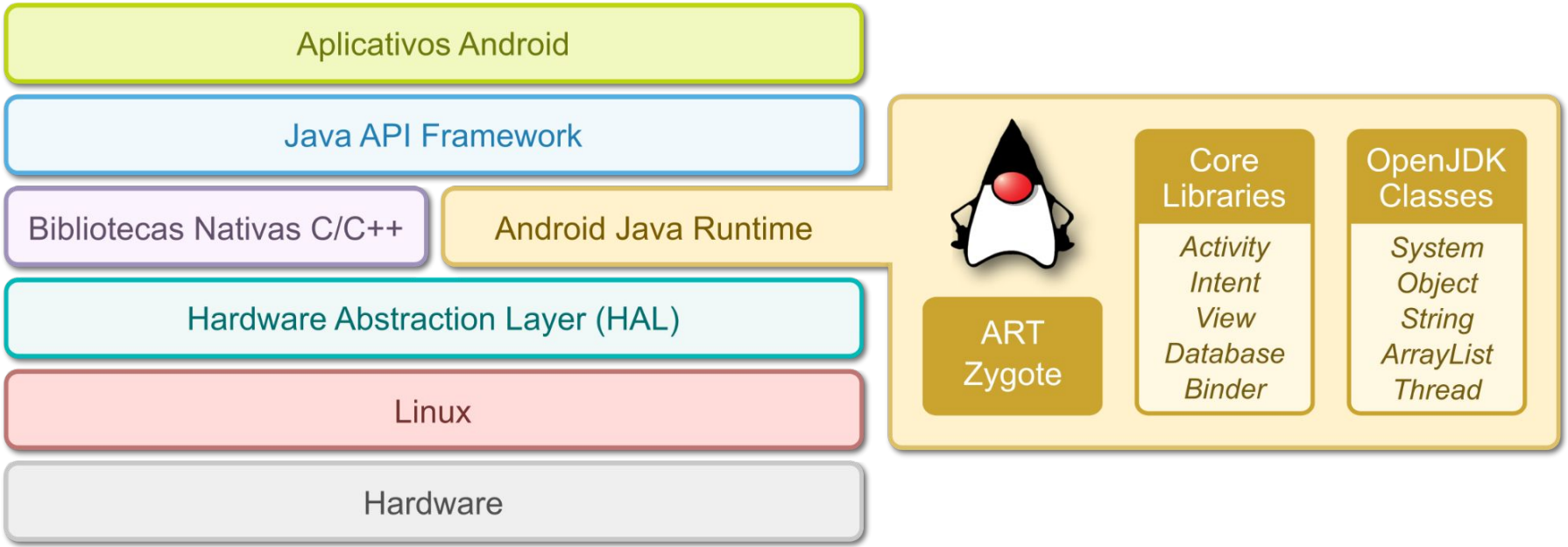
Camadas do Android



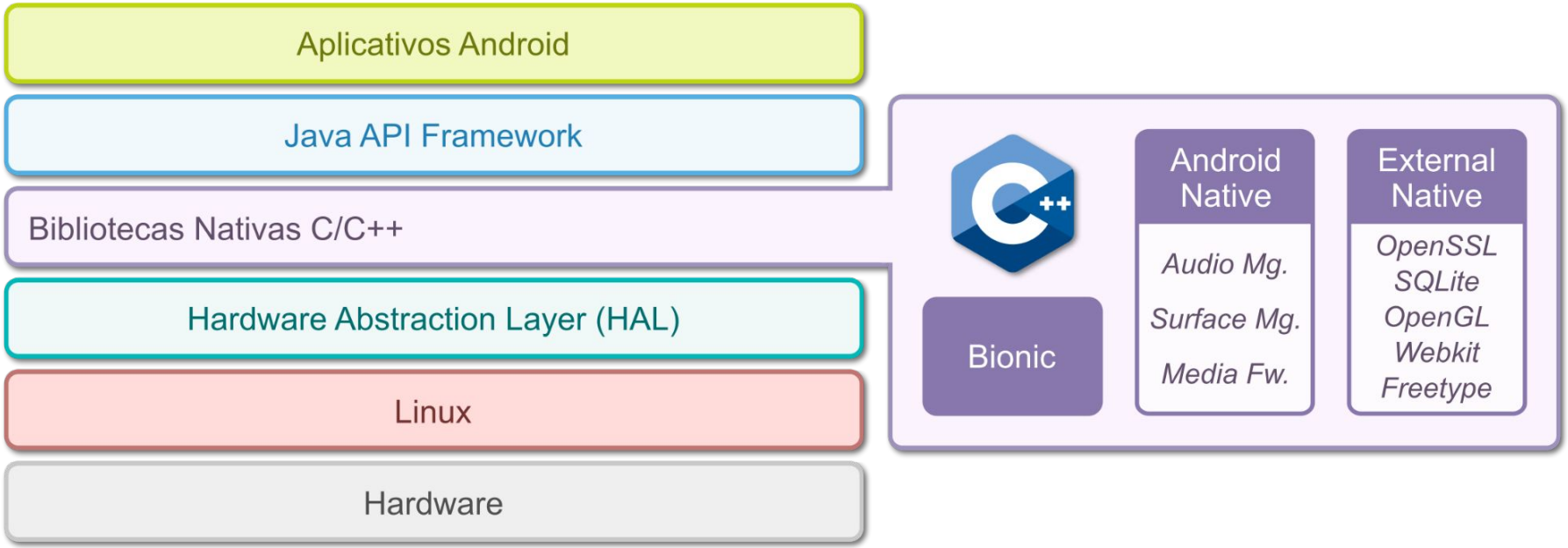
Camadas do Android



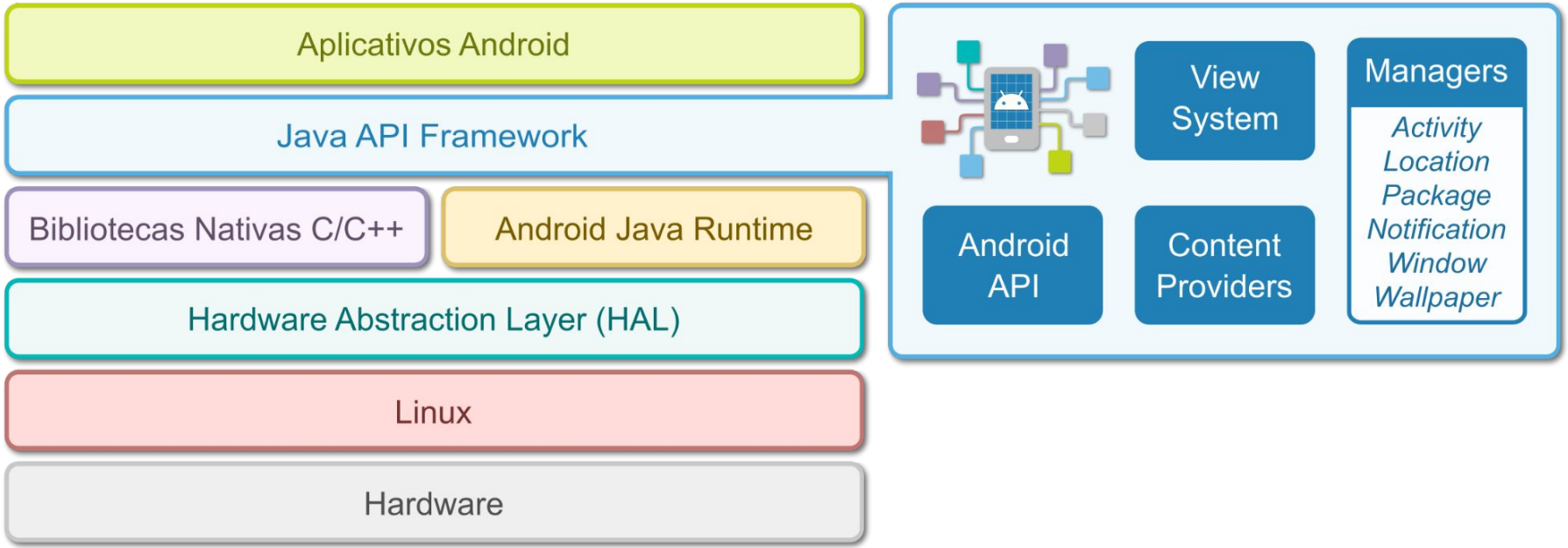
Camadas do Android



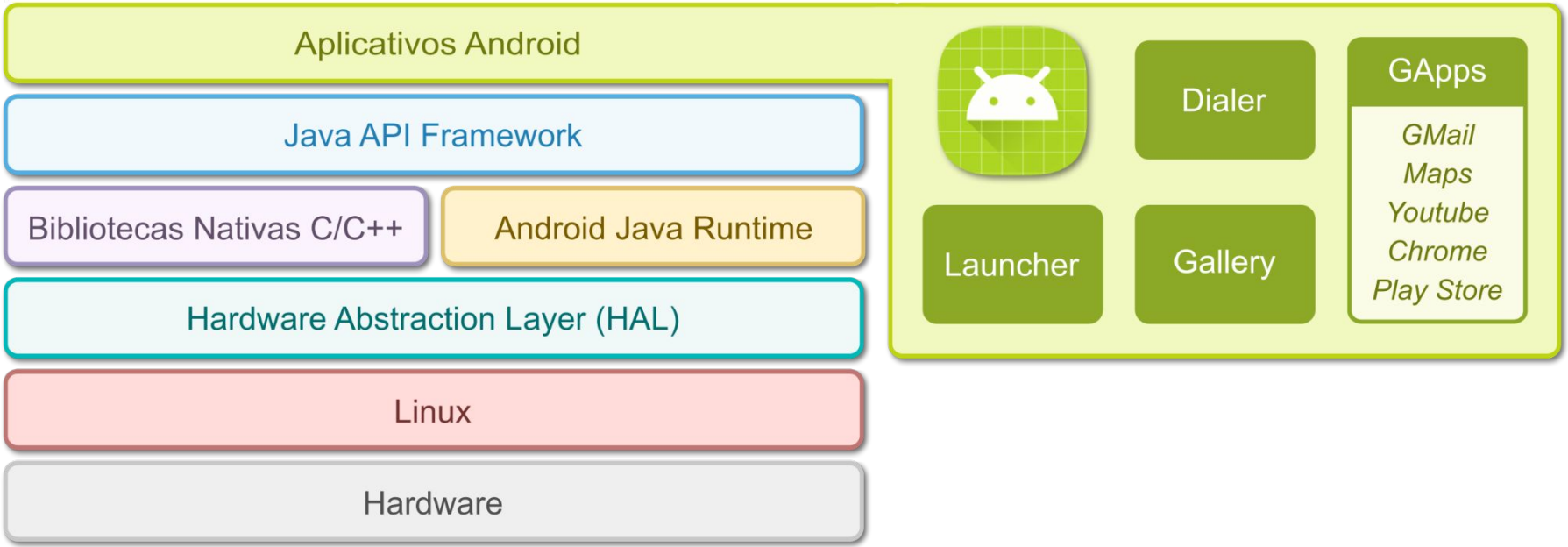
Camadas do Android



Camadas do Android



Camadas do Android

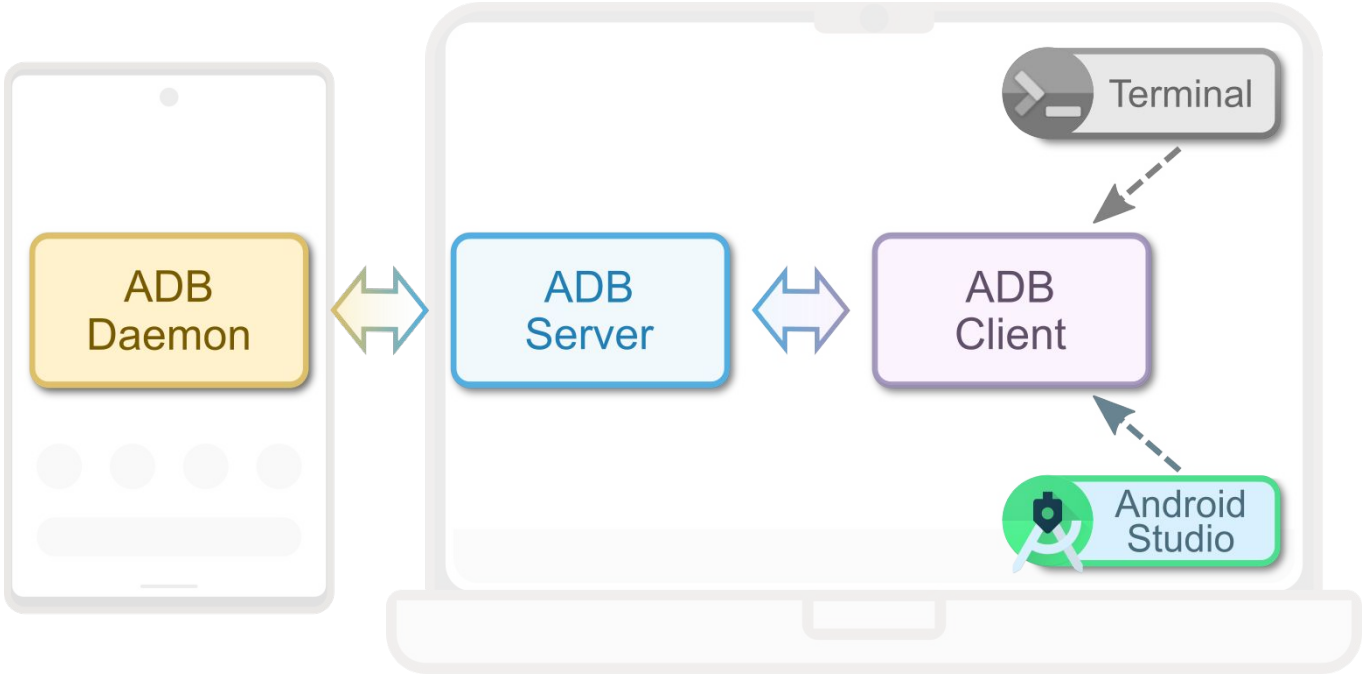


Android Debug Bridge

- Para acessar o Linux (terminal) do Android, vamos usar o ADB
- Android Debug Bridge – ADB
  - Programa de linha de comando
  - Permite a comunicação com o dispositivo Android
    - *seja ele um dispositivo real (e.g., smartphone) ou emulado*
- O ADB facilita ações como:
  - Instalar e desinstalar aplicações no smartphone
  - Depurar (debug) aplicações em execução
  - Execução de comandos
  - Transferência de arquivos
  - Acesso via shell

Android Debug Bridge

- Composto por três partes: *client*, *server* e *daemon*



Listando os Dispositivos

- Lista dos dispositivos conectados (reais e/ou emulados)

```
$ adb devices -l
```

Acessando o Terminal do Android

- Se conecta ao terminal do dispositivo

```
$ adb shell
```

Listando os Arquivos e Diretórios

- Arquivos e diretórios da raiz (/) do Android

```
$ ls
```

- Alguns diretórios são partições (alguns, só de leitura)

```
$ mount
```

Principais Diretórios

Diretório	Descrição
/system	Principal diretório do sistema do Android. Contém arquivos de configurações, arquivos de inicialização, bibliotecas e todo o framework do Android.
/product	Uma parte do sistema específica do produto atual. Contém aplicativos (Gallery, Camera, etc), arquivos de configurações, arquivos de inicialização, bibliotecas, e outros.
/vendor	Partição da fabricante do dispositivo (e.g., Motorola). Contém executáveis, programas, aplicativos, bibliotecas, drivers e configurações específicas do produto e do smartphone atual.
/system_ext	Extensões do sistema (framework) do Android implementadas pela fabricante. Tais extensões não fazem parte do AOSP. São criadas pela fabricante para seus dispositivos e aplicativos.
/data	Contém os dados e configurações do sistema e do usuário. Se essa partição for formatada, o dispositivo retorna à sua "configuração de fábrica". O diretório "/data/data" contém as configurações dos aplicativos.
/sdcard	Um link para "/storage/self/primary". Contém um local no dispositivo que o usuário e os aplicativos podem escrever com mais liberdade. Contém as fotos e os documentos, por exemplo.

Quem sou eu?

- Imprime o login do usuário atual

```
$ whoami
```

- Qual o arquivo com informações dos usuários?

```
$ cat /etc/passwd
```

- Os usuários de sistema são compilados junto com o Android:
  - Code Search: android_filesystem_config.h

Virando Root

- Virando root

```
$ su
```

Usuários como Primeira Linha de Defesa

- A primeira forma de defesa do Android é o Discretionary Access Control (DAC)
  - O controle de acesso é baseado em usuários, senhas e permissões
- Cada app instalado roda como um usuário diferente
  - Um pouco diferente do Linux tradicional, apesar de algo parecido ser usado pelos daemons/serviços de sistema (e.g., sshd, crond)
  - Isso evita que um app tenha acesso a dados de outros apps, pois os arquivos pertencerão a usuários diferentes
  - Aproveita o principal esquema de segurança já testado há várias décadas no Linux

Usuários como Primeira Linha de Defesa

- Abra o app Gallery no emulador e execute:

```
$ ps -ef | grep gallery
```

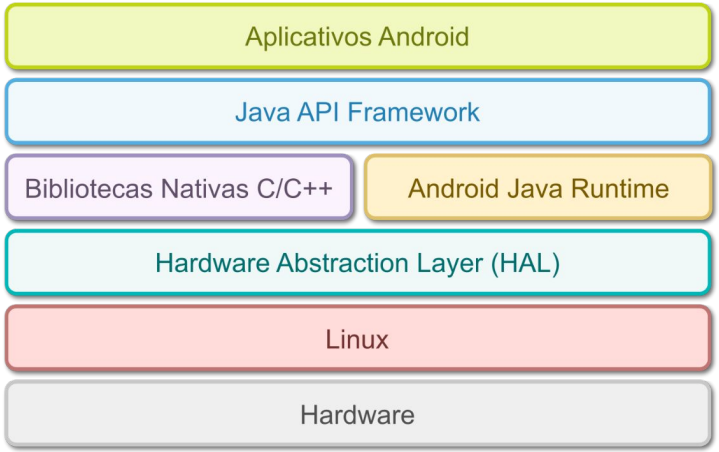
- Faça o mesmo para o app Contacts
  - Note como os usuários são diferentes

```
$ ps -ef | grep contacts
$ cd /data/data/com.android.contacts
$ ls -al
```

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.2: Compilando, Executando e Modificando o Android
  - Laboratório 4.3: Explorando o Android/Linux ADB, Sistema de Arquivos

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



Explorando o Android/Linux Parte II

Yuri Assayag
yfa@icomp.ufam.edu.br

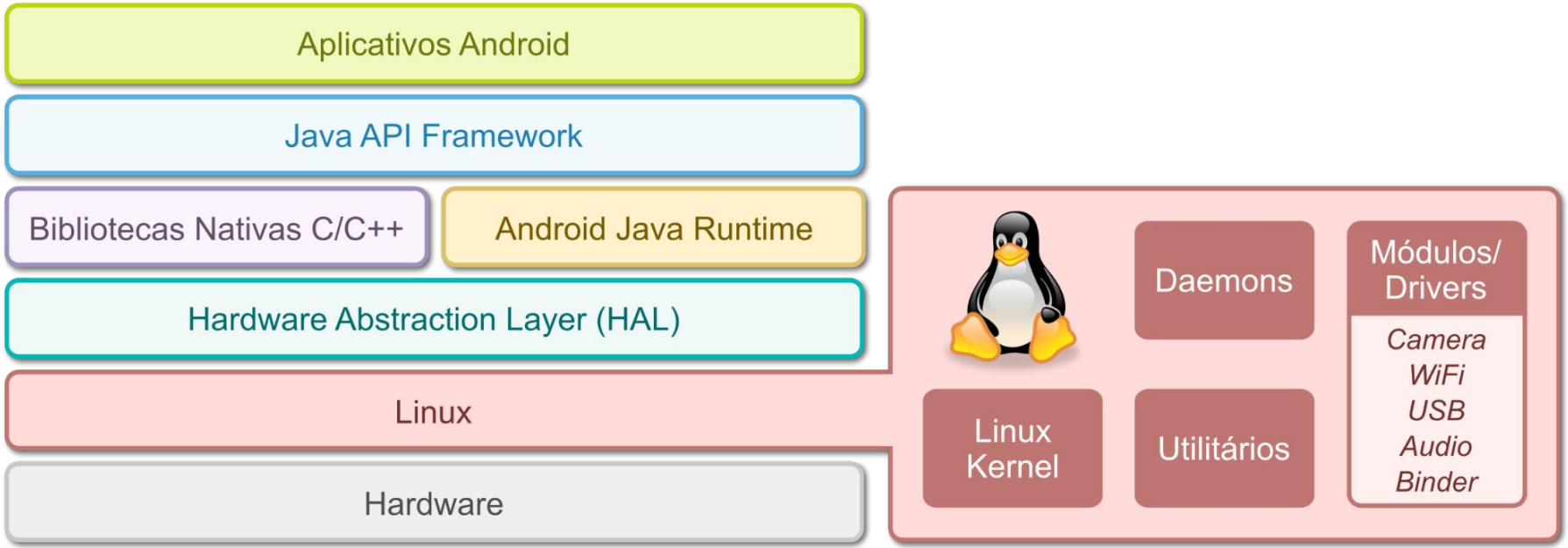
ColabWeb: bit.ly/devtitans-moodle

Configurando o Ambiente de Compilação

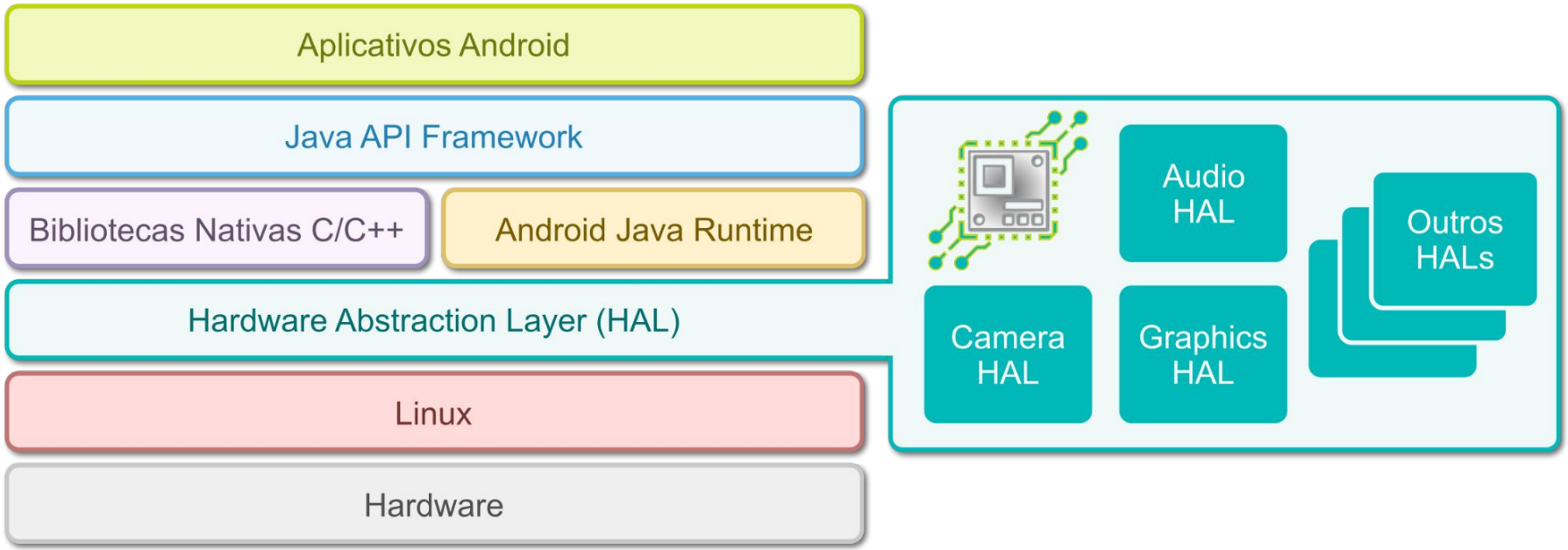
- Configuração do terminal e Compilação

```
$ cd ~/aosp
$ source build/envsetup.sh
$ lunch sdk_phone_x86_64-userdebug
$ m -j16
$ emulator
```

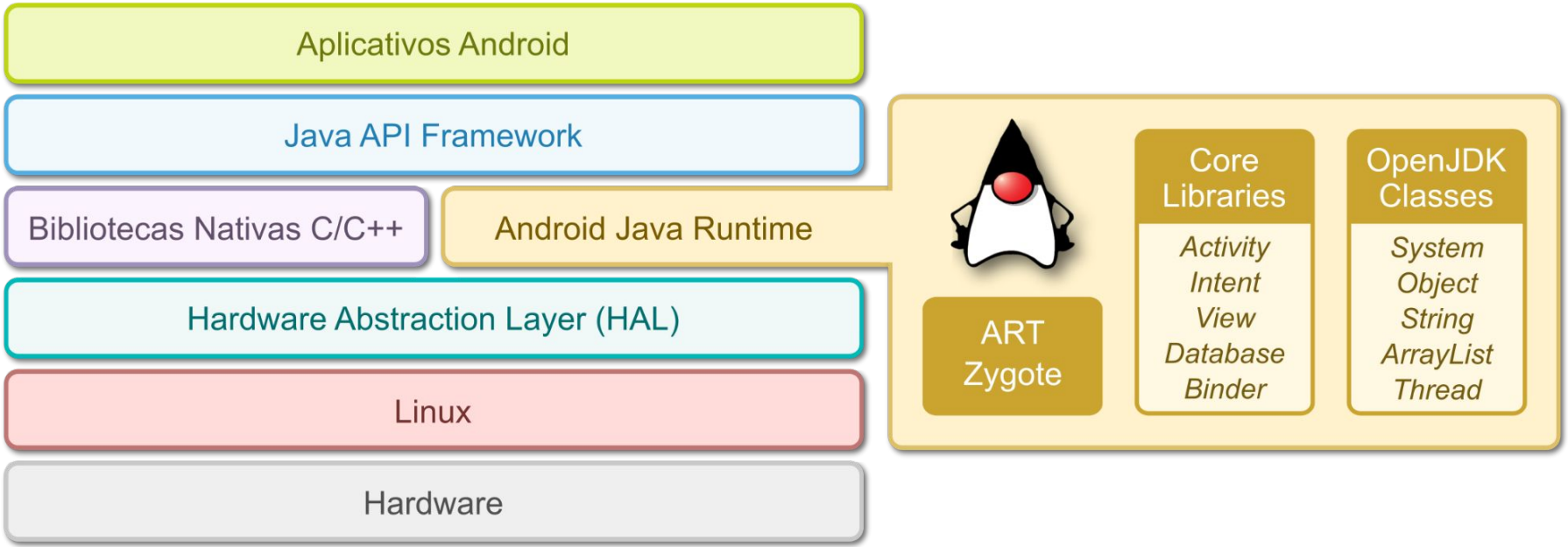
Camadas do Android



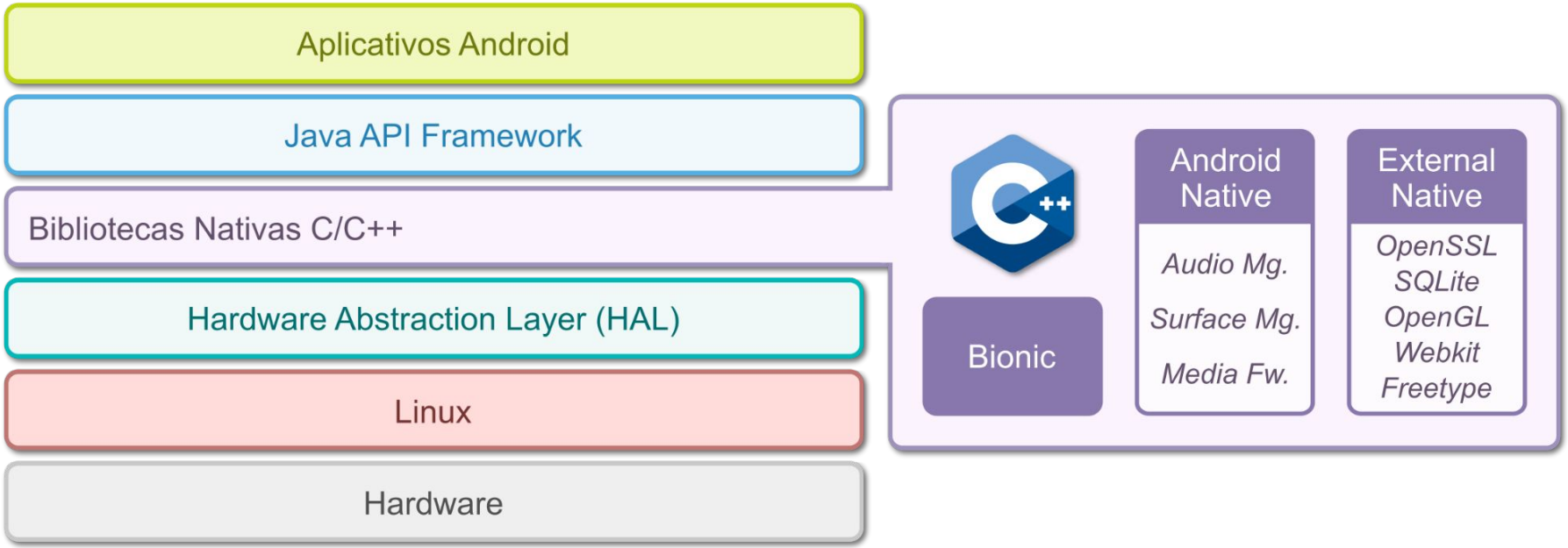
Camadas do Android



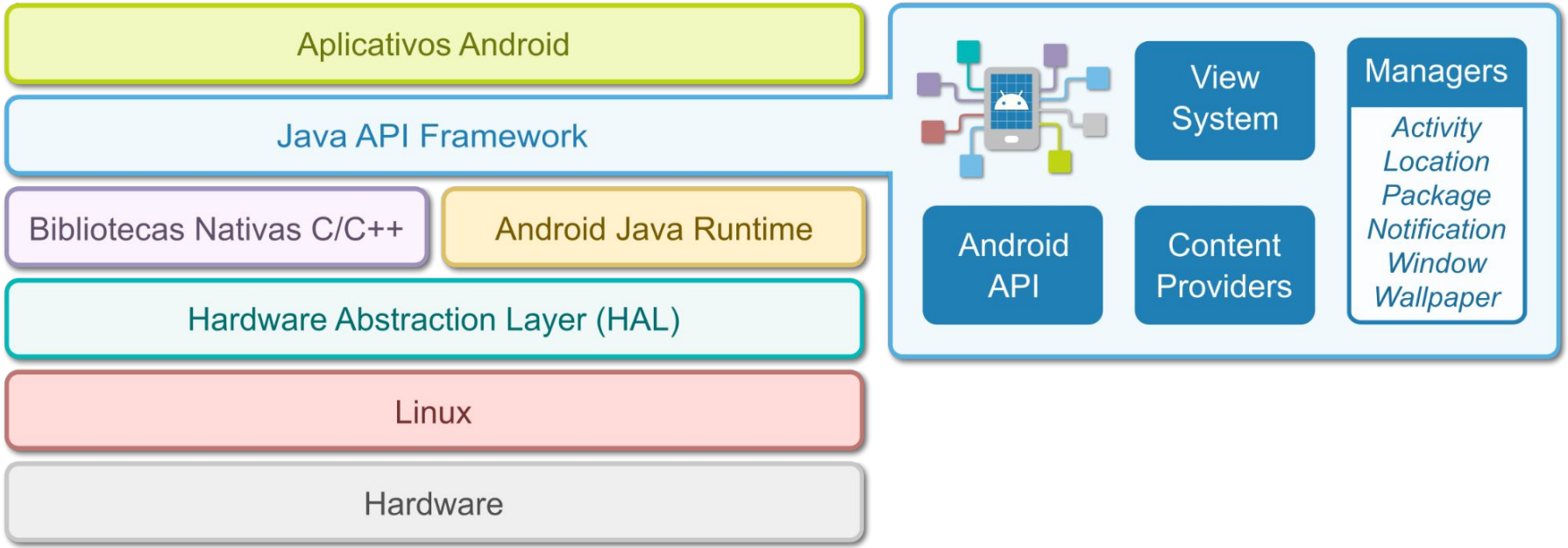
Camadas do Android



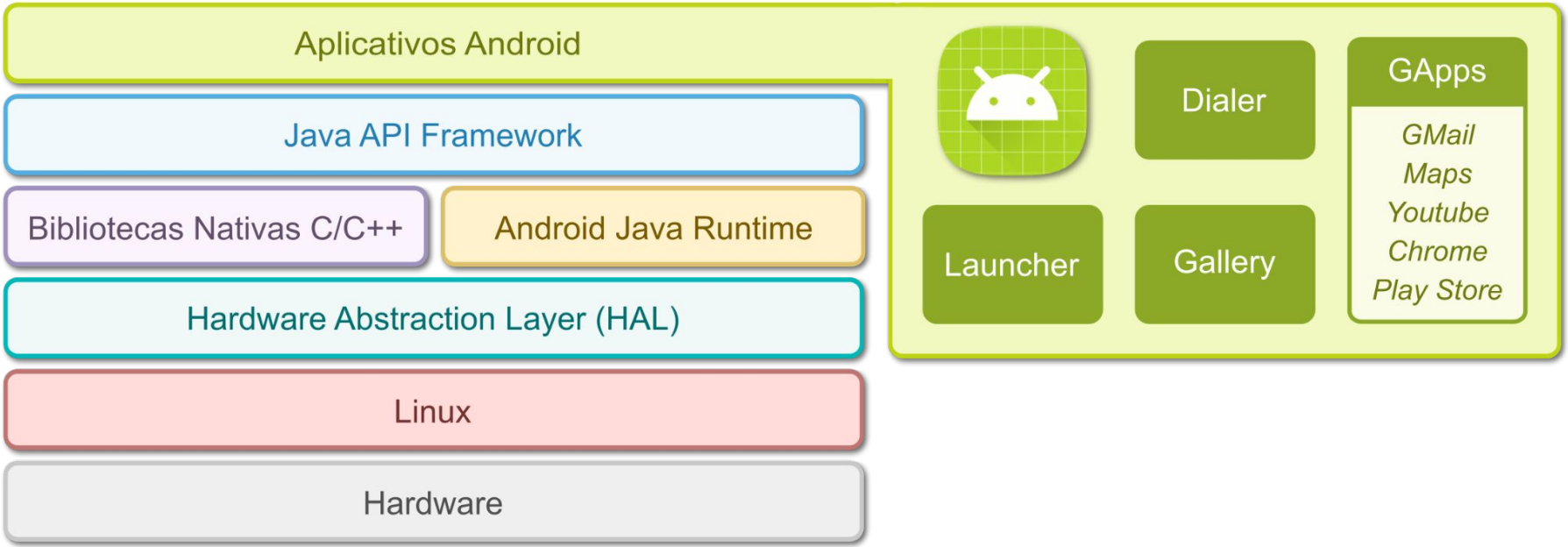
Camadas do Android



Camadas do Android



Camadas do Android



Acessando o Terminal do Android

- Se conecta ao terminal do dispositivo

```
$ adb shell
```

Principais Diretórios

Diretório	Descrição
/system	Principal diretório do sistema do Android. Contém arquivos de configurações, arquivos de inicialização, bibliotecas e todo o framework do Android.
/product	Uma parte do sistema específica do produto atual. Contém aplicativos (Gallery, Camera, etc), arquivos de configurações, arquivos de inicialização, bibliotecas, e outros.
/vendor	Partição da fabricante do dispositivo (e.g., Motorola). Contém executáveis, programas, aplicativos, bibliotecas, drivers e configurações específicas do produto e do smartphone atual.
/system_ext	Extensões do sistema (framework) do Android implementadas pela fabricante. Tais extensões não fazem parte do AOSP. São criadas pela fabricante para seus dispositivos e aplicativos.
/data	Contém os dados e configurações do sistema e do usuário. Se essa partição for formatada, o dispositivo retorna à sua "configuração de fábrica". O diretório "/data/data" contém as configurações dos aplicativos.
/sdcard	Um link para "/storage/self/primary". Contém um local no dispositivo que o usuário e os aplicativos podem escrever com mais liberdade. Contém as fotos e os documentos, por exemplo.

Linha de Comando

- Muitos dos principais comandos que você executa no Linux são programas diferentes
  - E.g., df, du, grep, cat
  - São implementados e mantidos por pessoas diferentes
  - Focam mais em funcionalidades e menos em ter tamanho pequeno
  - Quando todos esses programas são colocados em um sistema, eles tendem a ocupar muito espaço em disco e executar menos eficientemente

Busybox

- O Busybox (/system/xbin) é um conjunto de ferramentas de linha de comando (E.g., df, du, grep, cat).
  - Você não verá apps nativos do sistema necessitando de busybox, já que para instalar o busybox é necessário ter root ou bootloader desbloqueado
  - O Busybox raramente é adicionado nativamente em um sistema baseado em Android
  - Alguns apps que precisam de algumas das ferramentas do Busybox para funcionar perfeitamente: Titanium backup, Rom Toolbox, WPS WPA TESTER, e muitos outros.

Toybox

- Devido à limitação de recursos, sistemas embarcados como roteadores, câmeras e Android, **substituem** essas centenas de programas por um **único** programa
  - Que os implementa de forma simples, compacta e eficiente
  - Dentre os mais conhecidos, tem-se:
    - *BusyBox, muito usado em roteadores*
    - *Toybox, usado pelo Android, principalmente por motivos de licença de uso.*

Toybox

- Veja os binários do /system/bin:

```
$ cd /system/bin
$ ls -al
```

- Quantidade de comandos implementados no Toybox:

```
$ ls -al | grep toybox | wc -l
```

- Mostrar a lista de comandos

```
$ toybox
```

Sistemas de Logs do Android

- O Android possui dois sistemas de logs
  - Logs do kernel e do sistema Linux (*diagnostic messages*)
  - *Android Logging System*

Logs do Kernel e do Sistema Linux

- Funciona como no Linux

```
$ su
$ dmesg    # diagnostic messages
```

- Para ver mensagens com o tempo legível

```
$ dmesg -T
```

- Para ficar monitorando por novas mensagens

```
$ dmesg -T -w
```

Logs do Kernel e do Sistema Linux

- Escrevendo uma mensagem

```
$ echo "planet: Omicron Persei 8" > /dev/kmsg
```

- Arquivo /dev/kmsg

```
$ cat /dev/kmsg
```

Android Logging System

- É o sistema usado pelos apps e pelos diversos outros programas em Java ou C/C++ do Android
  - Nos programas em Java, a classe `android.util.Log` é utilizada
  - Já nos programas em C/C++, a biblioteca `liblog` é utilizada.
- Tudo é gerenciado pelo daemon logd
  - Processo responsável por centralizar todas as mensagens de logs do Android
- Para visualizar as mensagens, usa-se o utilitário logcat

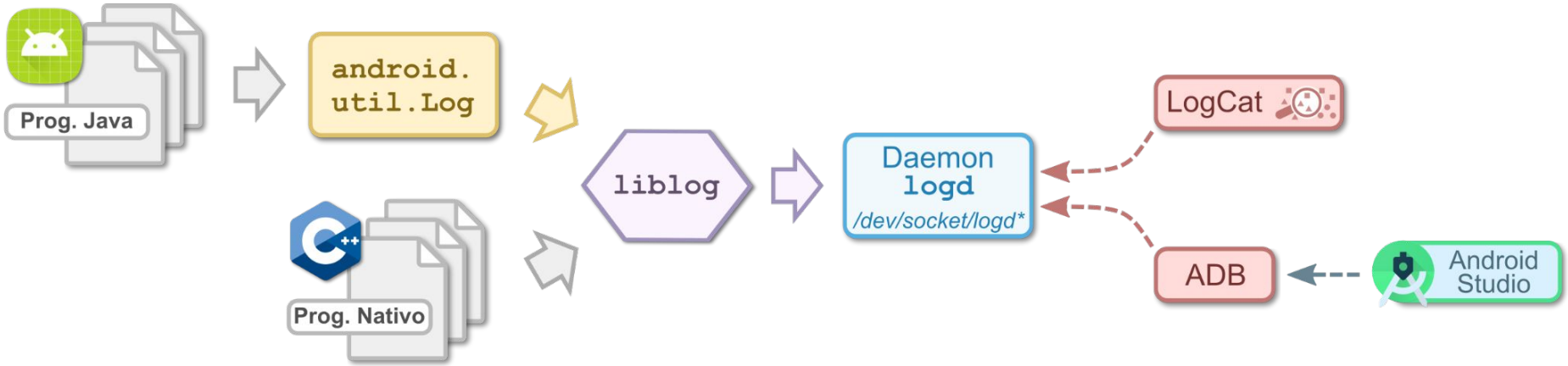
Android Logging System

- Prioridades/Níveis das mensagens:

Prioridade	Nome	Método Java	Descrição
V	Verbose	Log.v	Mensagens detalhadas de debug
D	Debug	Log.d	Mensagens normais de debug.
I	Info	Log.i	Mensagens de informações.
W	Warning	Log.w	Mensagens de alerta. Não é um erro, mas quase.
E	Error	Log.e	Uma condição de erro foi encontrada.

Android Logging System

- A figura a seguir resume o Android Logging System



Android Logging System

- Lendo as mensagens do logcat

```
$ logcat
```

Android Logging System

- Filtrando mensagens

```
$ logcat -s Zygote
```

- Filtrando por níveis

```
$ logcat -s *:I
```

Android Logging System

- Escrevendo uma mensagem

```
$ log -p I -t "MsgTeste" "Mensagem de Teste ..."
```

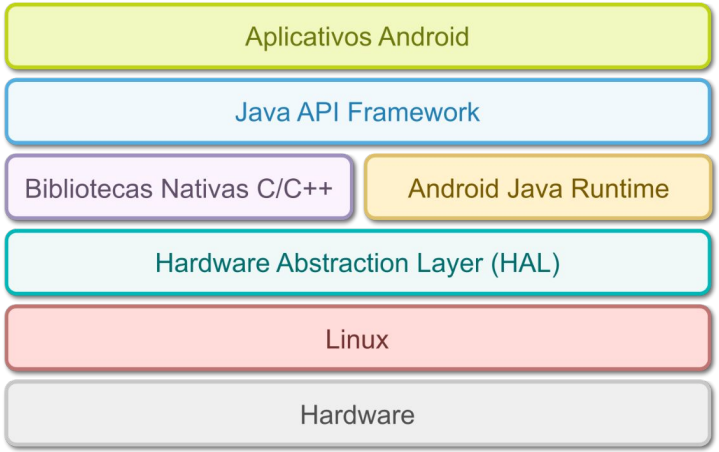
- Buscando a mensagem

```
$ logcat MsgTeste:I *:S
```

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.4: Explorando o Android/Linux - Toybox, Android Logging System

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



Explorando o Android/Linux Parte III

Yuri Assayag
yfa@icomp.ufam.edu.br

ColabWeb: bit.ly/devtitans-moodle

Eventos de Dispositivos de Entrada

- O comando `getevent` provê informações de eventos de dispositivos de entrada
  - Touchscreen
  - botões (e.g., power, volume)
  - pressionamento de teclas (mesmo virtuais), etc.

Lendo Eventos

- Lendo os eventos
  - Vá no emulador e pressione o botão de “abaixar volume”

```
$ getevent
```

- Lendo os eventos com os códigos decifrados

```
$ getevent -l
```

Escrevendo/Simulando Eventos

- O comando sendevent permite simular eventos. Sintaxe:
  - sendevent <device> <type> <code> <value>

```
$ adb shell "
$ su root sendevent /dev/input/event1 1 114 1
$ su root sendevent /dev/input/event1 1 116 1
$ su root sendevent /dev/input/event1 0 0 0
$ sleep 1
$ su root sendevent /dev/input/event1 1 114 0
$ su root sendevent /dev/input/event1 1 116 0
$ su root sendevent /dev/input/event1 0 0 0"
```

Propriedades do Sistema

- O Android possui propriedades de sistema, que podem ser acessados por programas, scripts e até mesmo apps
  - Permitem personalizar/modificar o comportamento de vários componentes da plataforma.
  - Podem ser vistas como uma espécie de “variáveis globais” do sistema

Lendo Propriedades do Sistema

- Para listar todas as propriedades:

```
$ getprop
```

- O nome da propriedade indica algumas características:
  - Propriedades que possuem o prefixo ro (read-only) são variáveis apenas para leitura e não podem ser modificadas.
  - Variáveis que possuem o prefixo persist são variáveis de escrita e leitura que mantém os valores mesmo depois que o sistema é reiniciado.
  - As outras variáveis que não possuem esses prefixos são variáveis de escrita e leitura que perdem o valor após o sistema ser desligado.

Lendo Propriedades do Sistema

- Mostrando o conteúdo de uma única propriedade

```
$ getprop ro.product.board
```

Criando Propriedades do Sistema

- Para criar uma nova propriedade

```
$ setprop palomakoba.pi "3.2546373698888"
```

Outros Comandos

- Capturar tela

```
$ screencap /sdcard/screenshot.png
```

```
$ screenrecord --size 1280x720 /sdcard/screenrecord.mp4
```

- Comandos do Linux (Toybox)

```
$ ps -ef
$ top
$ free -h
$ df -h
```

Daemons do Android

- Assim como o Linux, o Android possui diversos daemons que ficam em execução provendo algum serviço para o sistema
- A seguir, iremos listar alguns dos daemons específicos do Android

Daemons do Android

- Listando alguns daemons

```
$ ps -ef | grep adbd
```

Daemons do Android

Daemon	Descrição
adbd	É o ADB *daemon*, que fica esperando por conexões do ADB *server*, usado pelo ADB *client*.
init	O init é o primeiro programa chamado quando o Android inicia. Ele é chamado diretamente pelo kernel do Linux e é responsável por iniciar todos os outros programas/processos. Ele possui o PID 1.
installd	É um serviço que instala e gerencia pacotes de aplicativos (APKs). O início do código fonte dele é relativamente fácil de entender.
lmkd	*Android Low Memory Killer Daemon* (lmkd) é um processo que monitora a memória do dispositivo e, quando tem-se pouca memória livre, ele mata os processos menos essenciais. Devido a ele que não precisamos fechar aplicativos no Android.

Daemons do Android

Daemon	Descrição
logd	*Daemon* do *Android Logging System* (logd). É o sistema de logs usado pelos apps e pelos diversos outros programas em Java ou C/C++ do Android.
netd	O netd *daemon* executa a maioria das configurações de rede no Android. O comando ndc é usado para se comunicar com ele via linha de comando.
tombstoned	Este *daemon* é responsável por coletar e salvar informações de depuração sempre que um programa termina de forma inesperada. Os dados são salvos em /data/tombstones/.
ueventd	Ueventd gerencia eventos de dispositivos. Quando alguma mudança em um dispositivo acontece (e.g., o dispositivo é conectado), o kernel do Linux dispara o que é chamado de *uevent*. Este *daemon* é responsável por capturar essas chamadas. Ele gerencia também o diretório /dev e /sys.

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.5: Explorando o Android/Linux - Eventos, Utilitários, Daemons

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados

AOSP
Novo Produto

Yuri Assayag
yfa@icomp.ufam.edu.br

ColabWeb: bit.ly/devtitans-moodle

AOSP – Produto

- *Produto* é possivelmente um dos conceitos mais importantes para se criar um “novo Android” a partir do AOSP

- Um produto define as características e personalização do Android para ser incluído em um dispositivo/smartphone específico comercializado por uma fabricante

AOSP – Produto

Informações do Produto

- Nome do produto
- Dispositivo de destino
- Nome da marca
- Nome do modelo

Personalização do Produto

- Propriedades do sistema
- Aplicativos a serem instalados
- Bibliotecas a compilar e incluir no produto
- Arquivos a serem copiados para a imagem
- Sobreposição de configurações (*overlays*)
- Linguagens suportadas
- Animação de boot

Configurações de Hardware

- Arquitetura e variante do processador (CPU)
- Configurações de Componentes
- Linux Kernel, Drivers
- Partições

Produto – Arquivos Principais

- Três arquivos principais são usados para configurar um produto novo:
  - **`AndroidProducts.mk`**
    - *É o arquivo de entrada para a configuração do seu produto*
    - *O sistema de compilação do Android varre o diretório **device** procurando por estes arquivos*
    - *Adiciona o nome do seu produto nas opções do comando **lunch** e, em seguida, carrega outros arquivos .mk com configurações mais específicas (próximo arquivo)*
  - **`<nome_produto>.mk`**
    - *É o arquivo com as configurações mais específicas do seu produto. É chamado/incluído pelo arquivo anterior.*
  - **`BoardConfig.mk`**
    - *Contém configurações específicas do dispositivo físico (cpu, kernel, partições, SELinux, etc)*

Produto – Convenções de Nomes e Diretórios

- O AOSP possui uma série de convenções de nomes e diretórios
  - Facilitam o entendimento
  - Permite ao sistema de compilação encontrar os arquivos

- No caso dos produtos:
  - Os arquivos de configurações ficam em uma hierarquia dentro do diretório:
    - *<aosp_root>/device/*
  - Essa hierarquia segue o padrão:
    - *<nome_empresa>/<nome_dispositivo>/*

Nomes e Diretórios

```
$ cd ~/aosp
$ ls device
$ ls device/google
```

Configs. do Emulador

```
$ cd build/target/product/
$ nano sdk_phone_x86_64.mk
```

Produto – Variáveis

- A tabela a seguir descreve as principais variáveis necessárias para criar um produto novo:

Variável	Descrição
PRODUCT_BRAND	Nome do Android deste produto. Normalmente é simplesmente "Android".
PRODUCT_NAME	Nome do produto, tem que ser igual ao especificado no AndroidProducts.mk. Normalmente, possui o formato <nome_empresa>_<nome_dispositivo>.
PRODUCT_DEVICE	Nome do dispositivo-alvo deste produto. Normalmente é igual à segunda parte do nome do produto (<nome_dispositivo>).
PRODUCT_MODEL	Nome do modelo do dispositivo-alvo deste produto.

Criando um Produto

```
$ cd device/devtitans/kraken/
$ nano AndroidProducts.mk
$ nano devtitans_kraken.mk
$ nano BoardConfig.mk
```

Produto – Outras Variáveis

- Nos próximos assuntos, iremos mostrando outras variáveis que personalizam o AOSP. Segue uma visão geral de algumas delas:

Variável	Descrição
PRODUCT_PACKAGES	Instrui o sistema de compilação a compilar e instalar módulos do AOSP que incluem apps, programas nativos, bibliotecas, recursos, etc.
PRODUCT_COPY_FILES	Instrui o sistema de compilação do AOSP a copiar um arquivo para a imagem final do Android.
PRODUCT_VENDOR_PROPERTIES	Cria/atribui um valor para uma propriedade de sistema do Android.
PRODUCT_LOCALES	Linguagens suportadas pelo produto.
PRODUCT_SHIPPING_API_LEVEL	API level do Android suportada pelo dispositivo.
DEVICE_PACKAGE_OVERLAYS	Define um diretório para ser usado como overlay. Esse diretório contém arquivos, em geral XML, com configurações que substituem originais.

Compilando o Produto

```
$ lunch
$ lunch devtitans_kraken-eng
$ m
```

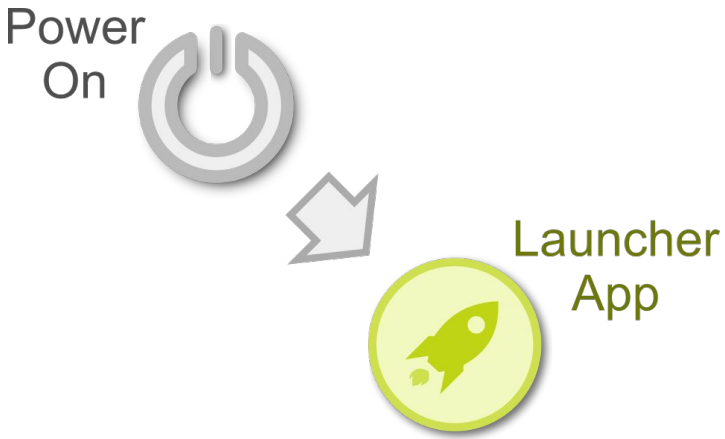
Executando o Produto

```
$ emulator
```

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.6: Criando um Novo Produto

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



AOSP – Personalizando a Inicialização do Sistema

Yuri Assayag
yfa@icomp.ufam.edu.br

ColabWeb: bit.ly/devtitans-moodle

Inicialização do Android

Power
On

Init – Primeiro Processo

- Conforme visto, o Init é o primeiro processo iniciado pelo Kernel
  - Possui PID 1
  - Responsável por iniciar os outros processos e configurar o sistema
  - Funciona de forma semelhante ao init do Linux

- O Init possui as seguintes funções:
  - Iniciar daemons
  - Configurar o sistema (propriedades, variáveis de ambientes, etc)
  - Iniciar o Zygote
  - Carregar e configurar módulos adicionais do Kernel
  - Setar permissões de arquivos

Init – Configurações

```
$ ls /system/etc/init/
```

- O Init é configurado através de arquivos localizados no diretório:
  - `/system/etc/init/`
    - *Tais arquivos estão no formato Android Init Language*

Init – Configurações

```
$ more /system/etc/init/hw/init.rc
```

- O primeiro arquivo de configuração lido pelo Init é o:
  - `/system/etc/init/hw/init.rc`
    - *Arquivo grande, mais de 1000 linhas*
    - *Importa outros arquivos .rc*
    - *Acessa propriedades do sistema. E.g.: ${ro.hardware}*

Android Init Language

- Android Init Language é o formato dos arquivos de configuração do Init
- São compostos por apenas 5 tipos de expressões:
  - Services, Actions, Commands, Options, Imports
- Um arquivo de configuração é dividido em uma ou mais seções
  - Uma seção pode ser do tipo Service ou do tipo Action
  - Dentro das seções podemos ter um ou mais Commands e Options
  - É só isso ... :)
  - Vamos ver um exemplo de Service

Service

```
$ cd /system/etc/init/
$ cat surfaceflinger.rc
$ ps -eo CMD,USER,GROUP | grep surfaceflinger
```

- A primeira linha é “service”:
  - Um serviço (daemon) será iniciado
- A segunda linha indica a classe do serviço
  - Serviços na mesma classe podem ser iniciados/parados juntamente
- Note o usuário e grupo do processo
- O *onrestart* indica que quando esse serviço for reiniciado, o serviço zygote precisa ser reiniciado também
  - Esquema de "dependência" entre serviços

Android Init Language – Action

- Como visto, uma seção pode ser do tipo *Service* ou do tipo *Action*
  - No slide anterior, vimos o *Service*, que inicia um daemon

- Um *Action* indica um conjunto de comandos que são executados a partir de um "gatilho"
  - Um gatilho, do inglês trigger, é disparado/executado sempre que o evento associado ao mesmo ocorrer
  - Quando esse evento do gatilho ocorrer, as ações (comandos) são executados

Action

```
$ cat /init.environ.rc
```

Action – Eventos

- Seguem alguns dos principais estados/estágios que o Init reconhece:

Estado	Descrição
early-init	Primeiro estágio de inicialização. Usado para configurações do sistema.
init	Monta as partições, configura variáveis do kernel e cria sistemas de arquivos.
early-fs	Executado pouco antes dos sistemas de arquivos estarem prontos para serem montados.
fs	Especifica as partições a serem montadas.
post-fs	Executado depois que as partições foram montadas, com exceção do /data.
post-fs-data	A partição /data foi descriptografada (se necessário) e montada.
boot	Comandos de boot normais.

Action – Comandos

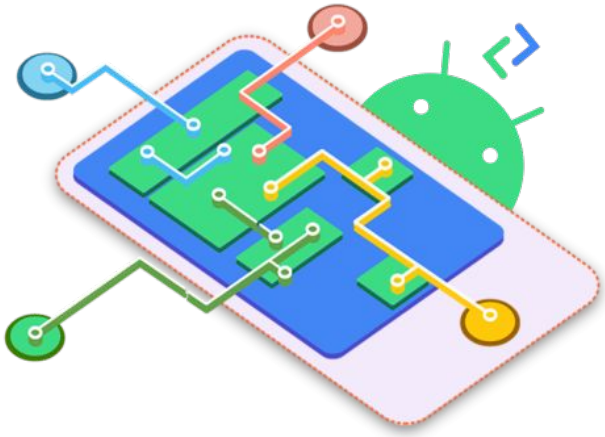
- Seguem alguns dos principais comandos que o Init reconhece
  - Alguns com o mesmo nome de comandos Linux, mas não são “binários”

Comando	Descrição
chmod, chown	Muda permissões de arquivos.
copy, rm, mkdir, rmdir	Manipulação de arquivos e diretórios.
exec	Executa um comando do sistema.
export	Seta uma variável de ambiente.
setprop	Seta uma propriedade do sistema.
insmod	Insere um módulo do Kernel do Linux.
start, stop, restart	Inicia, pára e reinicia um serviço do Init.

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.7: Sistema de Inicialização do Android

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



Personalizando Arquivos, Init e Propriedades

Yuri Assayag
yfa@icomp.ufam.edu.br

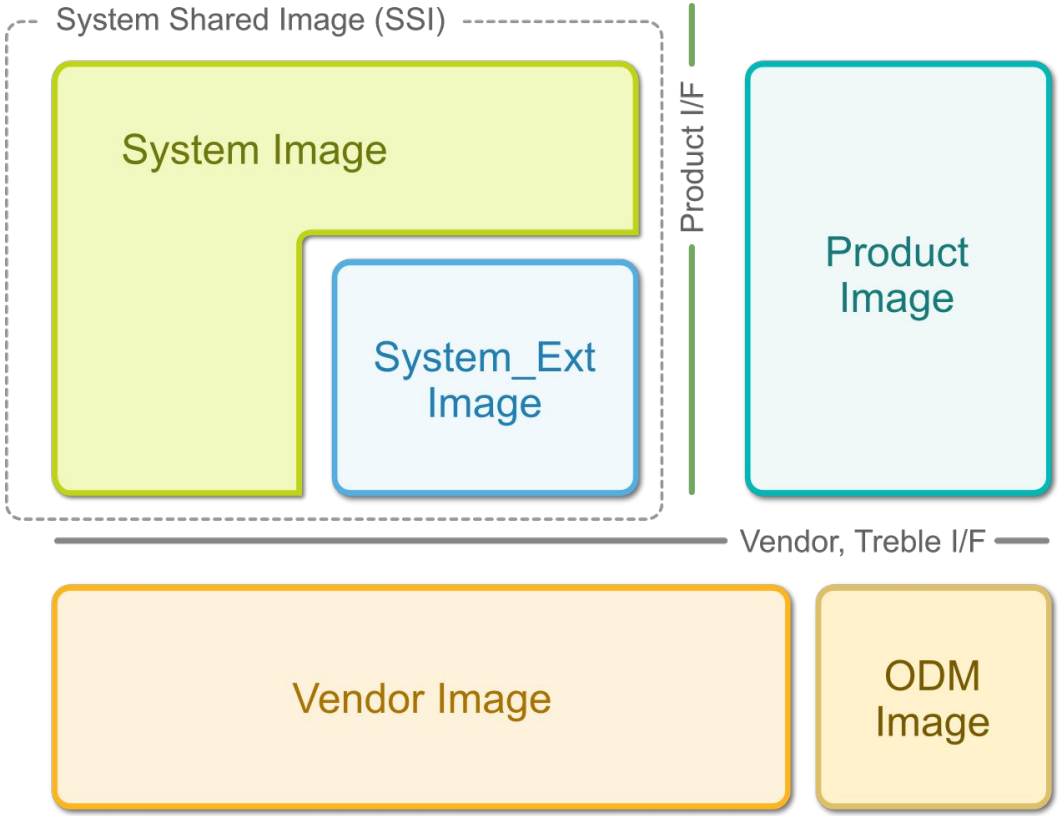
ColabWeb: bit.ly/devtitans-moodle

Partições Principais do Android

- Em aulas anteriores, vimos algumas partições principais do Android:

Partição	Descrição
/system	Partição principal do sistema. Contém o código-fonte original do AOSP. O ideal é que nada nessa partição seja alterado pelos fabricantes.
/system_ext	Extensões da fabricante ao sistema Android. Será igual em todos os dispositivos/produtos da mesma fabricante. Fortemente conectado ao system e são atualizados juntos.
/product	Personalizações (apps, configurações, etc) específicas de um único dispositivo/produto.
/vendor e */odm*	Personalização do hardware do dispositivo. Podem ser combinadas na partição vendor.

Partições Principais do Android



Copiando Arquivos para o Android

- Usa-se a variável PRODUCT_COPY_FILES
  - Dentro do arquivo devtitans_kraken.mk
  - Indica, ao sistema de compilação, os arquivos que precisam ser copiados do computador local para a imagem do Android (e para qual partição)

Copiando Arquivos para o Android

```
$ nano devtitans.txt
$ nano devtitans_kraken.mk
---
PRODUCT_COPY_FILES += \
    device/devtitans/kraken/devtitans.txt:system/etc/devtitans.txt
---
```

Modificando o Init

- Vamos agora usar nossos nossos conhecimentos do Init para personalizar a inicialização do sistema

Copiando Arquivos do Init

```
$ nano kraken.rc
---
on early-init
    export DEVTITANS "DevTitans Kraken! v1.0"

on property:dev.bootcomplete=1
    exec /system/bin/log -p d -t "DevTitans" "Kraken iniciado."
---


$ nano devtitans_kraken.mk
---
PRODUCT_COPY_FILES += \
    device/devtitans/kraken/devtitans.txt:system/etc/devtitans.txt \
    device/devtitans/kraken/kraken.rc:vendor/etc/init/kraken.rc
---
```

Verificando

```
$ logcat | grep DevTitans
$ echo $DEVTITANS
```

Modificando Propriedades do Sistema

- No módulo anterior, vimos como acessar/criar/alterar uma propriedade do sistema:
  - Usando os comandos *getprop* e *setprop* do terminal.

- Para fazer isso em tempo de compilação, usa-se a variável PRODUCT_SYSTEM_PROPERTIES
  - Dentro do arquivo devtitans_kraken.mk
  - Tem-se também as variáveis PRODUCT_PRODUCT_PROPERTIES e PRODUCT_VENDOR_PROPERTIES, que serão vistas no laboratório

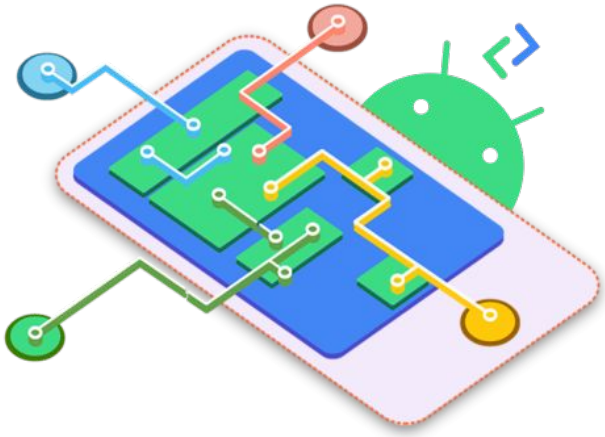
Modificando Propriedades do Sistema

```
$ nano devtitans_kraken.mk  # No final do arquivo
---
PRODUCT_SYSTEM_PROPERTIES += \
    ro.devtitans.name=Kraken
---
```

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.8: Personalização do Produto - Init, Propriedades

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em Android e Sistemas Embarcados



Personalizando Overlay, Background e BootAnimation

Yuri Assayag
yfa@icomp.ufam.edu.br

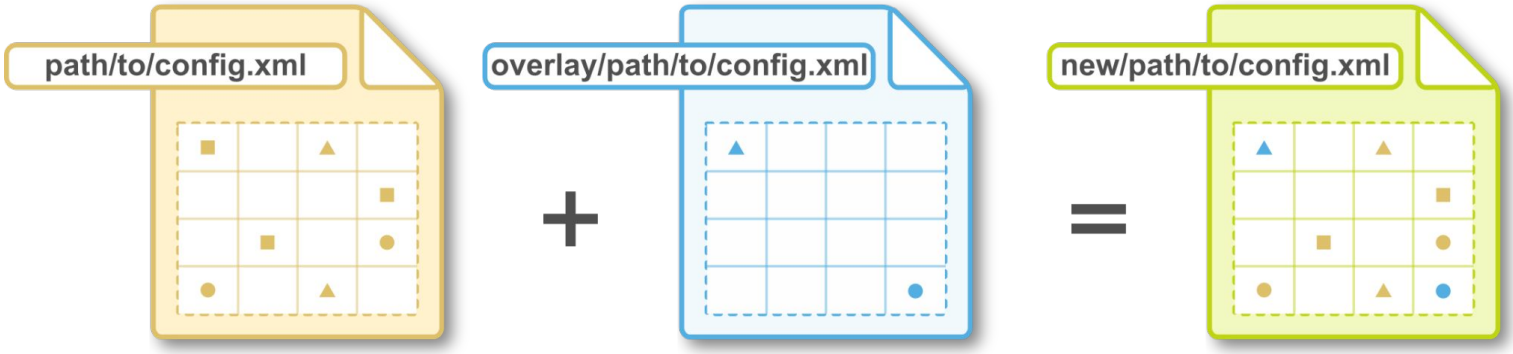
ColabWeb: bit.ly/devtitans-moodle

Overlay

- Permite "substituir" informações presentes em qualquer arquivo de recursos do AOSP
  - Em geral, "arquivos de recursos" são arquivos XMLs
  - Mas podem ser qualquer arquivo do diretório "res" (e.g., uma imagem)

- Basicamente, o overlay é um diretório no seu produto
  - O sistema de compilação busca por arquivos nele para personalizar/substituir os arquivos originais do AOSP

Overlay


path/to/config.xml
overlay/path/to/config.xml
new/path/to/config.xml

Overlay – Configurando o Uso

- Para configurar o uso do overlay, usamos a opção PRODUCT_PACKAGE_OVERLAYS

```
$ mkdir overlay
$ nano devtitans_kraken.mk
---
PRODUCT_PACKAGE_OVERLAYS = device/devtitans/kraken/overlay
---
```

Overlay – Modificando Configurações

- Um dos arquivos mais comuns de serem personalizados é o:
  - frameworks/base/core/res/res/values/config.xml

```
$ mkdir -p overlay/frameworks/base/core/res/res/values/
$ nano overlay/frameworks/base/core/res/res/values/config.xml
---
<resources>
    <dimen name="config_dialogCornerRadius">50dp</dimen>
</resources>
---
```

Overlay – Modificando o Wallpaper Padrão

- O overlay pode também ser usado para modificar outros arquivos além de arquivos XML
  - Por exemplo, a seguir iremos substituir o wallpaper inicial do Android

- O Wallpaper padrão é o arquivo:
  - frameworks/base/core/res/res/drawable-nodpi/default_wallpaper.png
  - Vamos substituí-lo usando o overlay

Overlay – Modificando o Wallpaper Padrão

```
$ mkdir -p overlay/frameworks/base/core/res/res/drawable-nodpi/
$ cp ~/default_wallpaper.png \
                overlay/frameworks/base/core/res/res/drawable-nodpi/
```

Modificando a Animação de Boot

- A animação de boot é comumente chamada de *boot animation*
  - Em geral, é o arquivo *product/media/bootanimation.zip*

- Como o arquivo de animação de boot não é um arquivo de recurso (diretório res), ele não pode ser substituído usando o Overlay.
  - Vamos simplesmente copiar um arquivo para substituí-lo

- O formato deste arquivo .zip será detalhado no laboratório

Modificando a Animação de Boot

```
$ cp ~/bootanimation.zip .
$ nano devtitans_kraken.mk
---
PRODUCT_COPY_FILES += \
    device/devtitans/kraken/devtitans.txt:system/etc/devtitans.txt \
    device/devtitans/kraken/kraken.rc:vendor/etc/init/kraken.rc \
    device/devtitans/kraken/bootanimation.zip:product/media/bootanimation.zip
---
```

Laboratório!

- Entre no Moodle:
  - https://bit.ly/devtitans-moodle
- Faça os laboratórios:
  - Laboratório 4.8: Personalização do Produto - Init, Propriedades
  - Laboratório 4.9: Personalização do Produto - Overlay, Boot Animation