Programação para Dispositivos Móveis

Leopoldo Teixeira
lmt@cin.ufpe.br | @leopoldomt

Até agora…

- Nossos apps consistem de apenas uma activity (em geral…)

- Normalmente aplicativos tem múltiplas activities, além de outros tipos de componentes

- Antes de avançarmos, é útil aprendermos como isto se relaciona com o SO

Processos:
de onde vem,
como nascem,
por quanto tempo vivem?

Bastidores

- Ao iniciar um app, Android faz um fork de um processo chamado de zygote

- O processo vai conter:
  - Cópia da VM (Dalvik ou ART)
  - Cópia das classes do Android Framework
  - Cópia das classes do app (via APK)
  - Objetos criados a partir das classes de framework, como as instâncias das subclasses de Activity

Suponha que seu aplicativo tem apenas uma activity.

O usuário vai abrir o app a partir do launcher...

Uma vez que esteja rodando, o usuário aperta o botão BACK.
O que acontece?

Android tenta manter seu processo vivo, ao menos por um tempo, por qual razão?

E se o usually aperta o botão HOME...
O que acontece?

A diferença é o que acontece com a Activity...

BACK vs. HOME

- Ao pressionar BACK, a activity em primeiro plano é destruída.

- A instância não será mais usada e fica marcada para ser *garbage collected*

- Ao pressionar HOME, a activity não é destruída imediatamente, mas permanece na memória

- Se o usuário retornar ao app, o sistema usa a instância existente ao invés de criar novo objeto

Termination

- Processos não vivem eternamente, pois ocupam espaço de RAM
- Eventualmente, Android tem de se livrar de processos para liberar memória
- O quanto seu processo vai durar depende de vários fatores, como por exemplo?

Fatores

- O que o dispositivo está fazendo em primeiro plano (aplicativos visíveis) e em segundo plano
- Quanto de memória o dispositivo tem
- O que ainda roda no processo

No cenário anterior, a diferença entre HOME e BACK é o potencial tempo que o processo ainda viverá…

Android tende a manter processos por mais tempo se estes tem componentes ativos.

Primeiro plano

- Android não elimina processos arbitrariamente
- Processos de primeiro plano são os últimos a serem eliminados (veremos mais detalhes)
- Se isso acontece, a coisa está feia… :)

Heap Size

- Processos usam RAM, e uma porção significativa será usada pelos objetos criados (heap)
- Os tamanhos de heap em Android não são grandes, e não controlamos usando **`-Xmx`**
- É imperativo ter isto em mente enquanto programamos…
- Para aplicativos simples como os que estamos vendo até agora, não há o que se preocupar

Activities

Activity

- Interface para interação com o usuário
- Cada *activity* deve dar suporte a apenas uma única tarefa do usuário
- Aplicações geralmente consistem de várias *activities*, cada uma com um propósito
- Isso significa que precisamos ter meios para iniciar *activities*

Vida,
Morte,
e Activities

Estados

- Active: activity foi iniciada, está em memória e em primeiro plano
- Paused: activity foi iniciada, está em memória, (parcialmente) visível, mas outra activity ocupa o primeiro plano, não é possível interagir
- Stopped: activity foi iniciada, está em memória, mas não é mais visível, por conta de activities que foram iniciadas
- Dead ou non-existent: activity nunca foi iniciada ou foi destruída, ou seja não está mais em memória

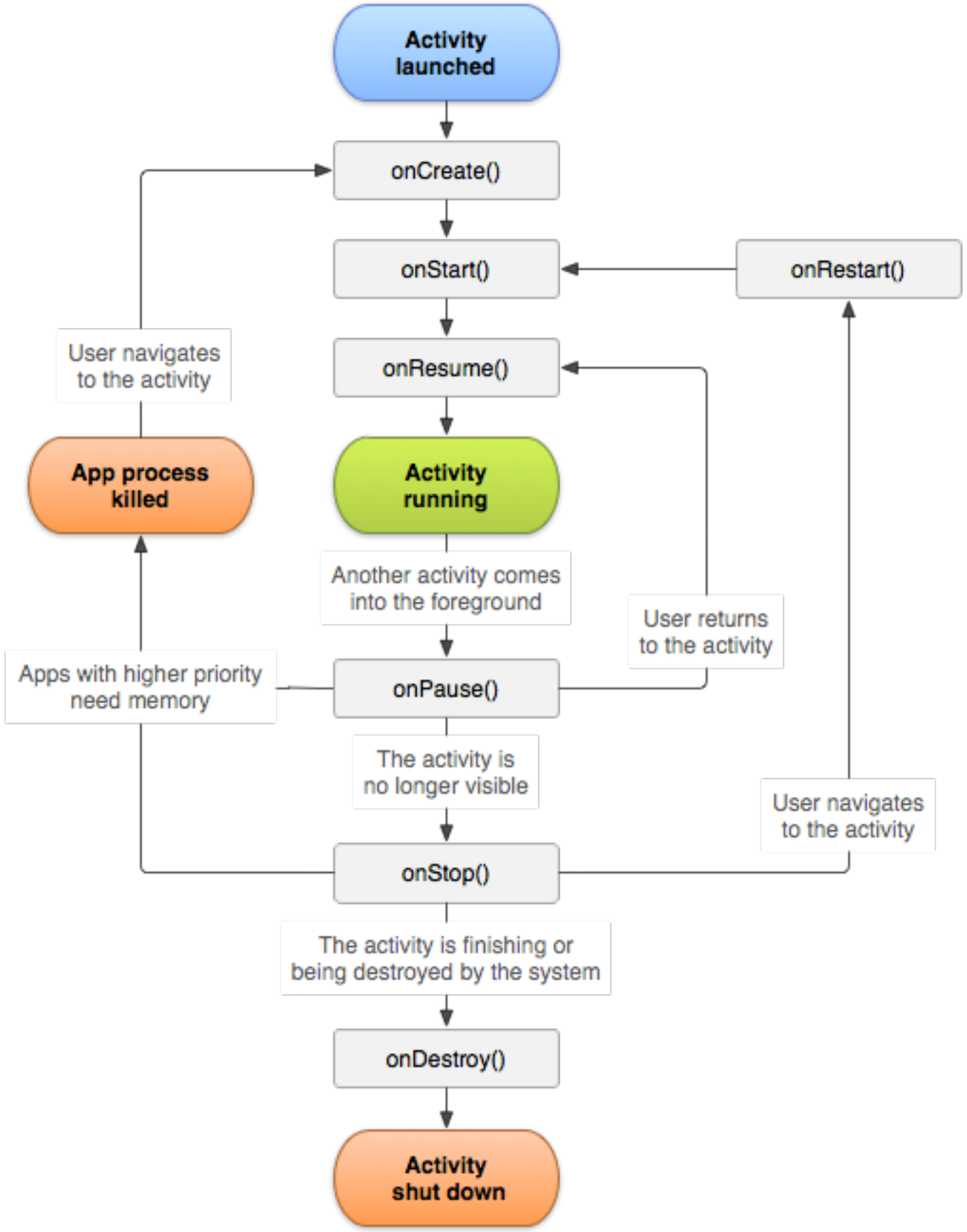
Ciclo de Vida

- Activities são criadas, suspensas, resumidas e destruídas conforme o necessário, durante execução da aplicação
- Algumas ações dependem do usuário
- Outras ficam a cargo do sistema

Métodos para mudança de estados

- Android fornece métodos para anunciar a mudança de estado em Activities

- De acordo com a necessidade da aplicação


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

`onCreate()`

- Chamado ao criar Activity
- Define estado inicial
  - pode definir layout (**`setContentView…`**)
  - inicializa referências para elementos de UI
  - configura interface conforme necessário

`onDestroy()`

- Activity está prestes a ser destruída, por conta do botão BACK, ou chamada a **`finish()`**

- Liberar recursos

- Pode não ser chamado se

  - ocorrer um crash com exceção não tratada

  - usuário usa *force-quit* via *Settings*

  - Android matar sua aplicação para liberar memória

`onStart()`

- Activity está prestes a se tornar visível
  - por estar sendo iniciada pela primeira vez
  - voltando ao primeiro plano após um tempo
- Iniciar comportamento que deve estar pronto quando aplicação se tornar visível
- Carregar dados da aplicação (persistência)

`onRestart()`

- Chamado se Activity foi pausada e está prestes a ser iniciada novamente

- Útil para processar ações que são necessárias apenas quando a aplicação foi pausada e reiniciada

`onStop()`

- Chamado assim que Activity não é mais visível para o usuário
  - Pode ser reiniciada mais tarde
- Cache state
- Pode não ser chamado pelas mesmas razões que **`onDestroy()`**

onResume()

- Activity está visível e prestes a iniciar interação com o usuário em primeiro plano

- Iniciar comportamento visível, *refresh* da UI

onPause()

- Prestes a perder o foco, como por exemplo, ao iniciar outra activity, ou ao aparecer um dialog
- Interromper comportamento visível
- Salvar estado

Trabalhe com os pares

- Se inicializou algo com X, limpe com Y
  - **x = onCreate(); y = onDestroy();**
  - **x = onStart(); y = onStop();**
  - **x = onResume(); y = onPause();**

Quando Activities morrem?

Quando Activities morrem?

- Ao pressionar o botão BACK
- Chamando **`finish()`** a partir da própria activity
  - Evite definir botões de “sair”/“*exit*”/“*quit*”
  - use as opções de Android, como botão BACK

Quando Activities morrem?

- Se nenhuma activity de um app está em primeiro plano, o processo é candidato para ser eliminado
  - pode ser que não execute **`onDestroy()`**
- Se o dispositivo passa por mudança de configuração, como por exemplo, orientação
- Se ocorrem exceções não tratadas

Iniciando *Activities*

Dois cenários

- Sabemos exatamente que *activity* desejamos iniciar
    - em geral, outra *activity* do mesmo app
- Temos uma referência para algo, como por exemplo, um link para página da web
    - queremos permitir que o usuário visualize, mas não sabemos exatamente quais opções temos disponíveis…

Explicit Intents

- Um Intent encapsula uma requisição, feita ao sistema, para algum componente fazer algo
- Se sabemos exatamente o que queremos, usamos intents explícitos, nomeando o componente que desejamos carregar

```
new Intent(this, ListActivity.class);
```

Implicit Intents

- Intents explícitos funcionam bem quando sabemos exatamente que componente utilizar

- No entanto, podemos iniciar activities a partir do SO ou de aplicativos de terceiros

- Nestes casos não temos um objeto **`Class`** que representa a Activity…

- Usamos a estrutura de intents implícitos, que se parecem com requisições Web (URIs)

Exemplos

- Imagine que você obteve latitude e longitude a partir de uma mensagem…
  - Podemos desejar exibir um mapa
    - Uma opção é embutir Google Maps…
    - A outra é simplesmente terceirizar para o sistema
      - “Ei, Android, abre aí uma activity que consegue exibir mapas para estas coordenadas…”

Exemplo

Explicit vs. Implicit

`Uri.parse()`

- No exemplo, vimos como usar este método para fazer o parsing de uma URL http(s)

- Existem outros schemes que podem ser usados

  - **`file://`** - via **`Uri.fromFile()`**

  - **`content://`** - content providers

`startActivity()`

VS.

`startActivityForResult()`

Navegação entre Activities

- Tasks
- Task Backstack
- Suspendendo e resumindo Activities

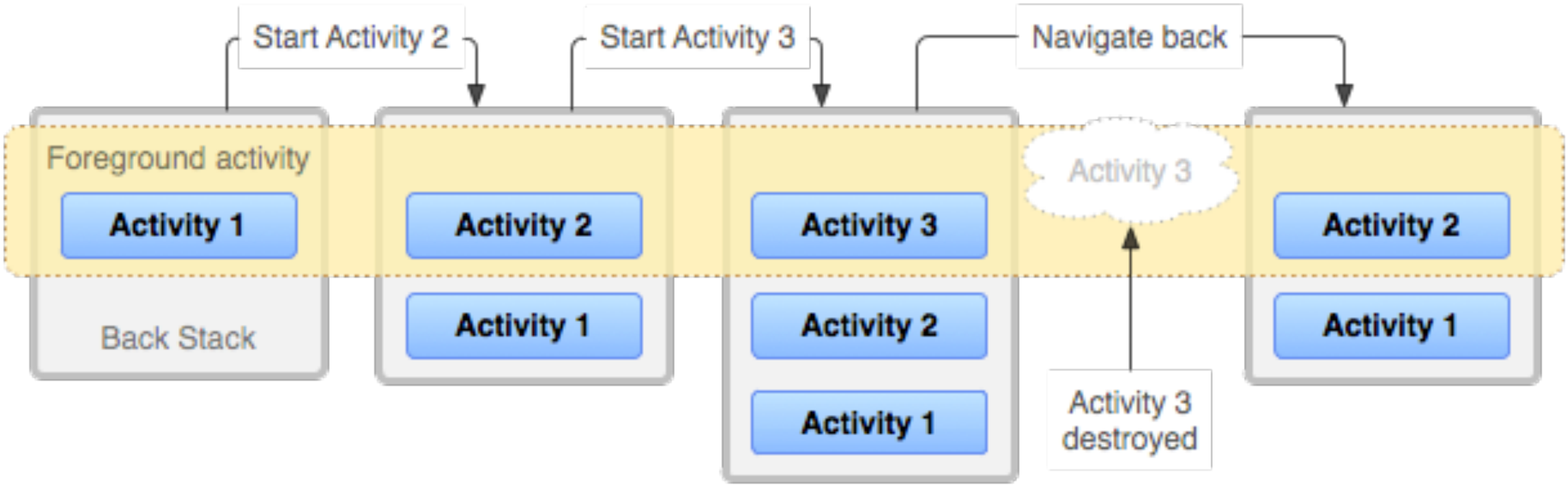
Tasks

- Um task é um conjunto de Activities relacionadas
- Estas activities não precisam ser parte da mesma aplicação
- Geralmente iniciados a partir de um clique no ícone do app

Task Backstack

- Quando iniciamos uma Activity, esta reside no topo da pilha
- Quando a Activity é destruída, ela é removida (pop) da pilha

Activities e Task Backstack



Processos

Iniciando componentes

- Quando um componente inicia e a aplicação não tem nenhum componente sendo executado, o sistema inicia um novo processo

- Por padrão, todos os componentes de uma mesma aplicação rodam no mesmo processo e thread (chamada de ‘main thread’)

Processos em Android

- As entradas do manifest de cada um dos tipos de componentes (**`<activity>`**, **`<service>`**, **`<receiver>`**, **`<provider>`**) possuem um atributo **`android:process`**, que permite especificar o processo que o componente irá rodar

- É possível fazer com que cada componente rode em um processo separado, ou fazer com que apenas alguns compartilhem o processo

- Também é possível fazer com que componentes de diferentes aplicações compartilhem o mesmo processo

Matando Processos

- Android pode decidir matar um processo, quando há pouca memória e há necessidade de uso da memória por outros processos

- Ao matar um processo, todos os componentes da aplicação rodando no processo são destruídos

- Para decidir que processos serão desligados, o sistema leva em consideração a importância relativa do processo para o usuário

Ciclo de vida

- Android tenta manter um processo rodando pelo maior tempo possível, mas eventualmente precisa remover processos antigos para liberar memória

- Para determinar quais processos devem ser mantidos e quais devem ser eliminados, há uma ‘hierarquia de importância’

- Processos menos importantes são eliminados primeiro, e assim sucessivamente, até atingir o necessário

Níveis de Importância

- Foreground
- Visible
- Service
- Cached

Foreground

- Um processo que está associado com o que o usuário está fazendo atualmente. Um processo é considerado foreground se alguma das condições abaixo for satisfeita:

  - Hospeda uma **`Activity`** que o usuário está interagindo no momento

  - Hospeda um **`Service`** que está executando um dos métodos de *callback* do ciclo de vida (**`onCreate()`**, **`onStart()`**, ou **`onDestroy()`**)

  - Hospeda um **`BroadcastReceiver`** que está executando o método **`onReceive()`**

Foreground

- Geralmente, apenas alguns processos estão neste nível em um dado momento

- Só são eliminados como última manobra - memória está tão baixa que não podem continuar a ser executados

- Geralmente, neste ponto, o dispositivo está em estado de paginação de memória, e precisa eliminar alguns processos para manter a interface com o usuário ‘responsiva’

Visible

- Processo que faz algo que o usuário esteja ciente.
  - Hospeda **`Activity`** que não está em primeiro plano, mas ainda está visível (**`onPause()`** foi chamado)
  - Hospeda **`Service`** que está rodando como *foreground*
  - Hospeda **`Service`** usado para uma feature que o usuário está ciente, como um *live wallpaper*... etc
- Um processo visível é considerado importante e não é eliminado, a não ser que isto seja necessário para manter os processos de primeiro plano rodando.

Service

- Um processo que roda um **`Service`** que foi iniciado com **`startService()`**, mas não se encaixa nas categorias anteriores

- Embora os processos de serviços não estejam diretamente ligados com algo que o usuário vê, geralmente estão fazendo algo que o usuário se importa, como um download

- Quanto mais tempo rodando, a importância é reduzida

Cached

- Estes processos não tem impacto direto na experiência do usuário e o sistema pode eliminá-los a qualquer momento para liberar memória

- Em um sistema funcionando normalmente, estes são os afetados pelo gerenciador de memória

- Tendência é manter múltiplos processos em cache e regularmente eliminar os mais antigos

Cached

- Um processo que hospeda uma **`Activity`** que não é visível ao usuário (método **`onStop()`** foi chamado)

- Se uma **`Activity`** implementa o ciclo de vida corretamente, e salva o estado atual, eliminar o processo não tem efeito visível para o usuário.

- São mantidos em uma (pseudo)lista de processos recentes (LRU), para garantir que o processo associado com a **`Activity`** mais recentemente vista pelo usuário seja o último a ser eliminado

Hierarquia de Processos

- Android tenta classificar processos com o maior nível possível, baseado na importância dos componentes atualmente ativos no processo. Por exemplo, se um processo hospeda um **`Service`** e uma **`Activity`** visível, o processo é classificado como Visible, ao invés de Service.

- Além disso, o nível de um processo pode ser aumentado pelo fato de outros processos serem dependentes do mesmo - um processo que está servindo a outro processo não pode ser definido com nível inferior ao do processo servido

Hierarquia de Processos

- Processos que rodam **`Services`** são superiores a processos com **`Activities`** de background

- Portanto, qualquer operação que tome mais tempo deve iniciar um **`Service`**, principalmente se for durar mais que a **`Activity`**

- Usar **`Services`** garante que pelo menos há uma certa prioridade do processo, independente do que acontece com a **`Activity`**

- Esta também é a razão pela qual **`BroadcastReceivers`** devem iniciar **`Services`** ao invés de realizar operações que demandam muito tempo

Programação para Dispositivos Móveis

Leopoldo Teixeira
lmt@cin.ufpe.br | @leopoldomt