Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

Trabalho Sistemas Operacionais

Docente: Diego Ferreira dos Santos

Discentes:

Rodrigo Luis Tavano Bosso – PC3005623

Rafael Abreu Coelho - PC3004481

Danylo Expedito de Sabino _ PC3005674

Sistema Operacional Android

0-Resumo:

O seguinte trabalho tem como objetivo explicar o funcionamento do sistema operacional Android 4.4, também conhecido como KitKat. Essa produção trabalhará tanto algumas características mais genéricas dos aparelhos com o Android; retratando, para isso, parte da história desse além dos demais SOs criados; quanto aspectos específicos desse, retratando nesse ponto parte dos funcionamentos para o gerenciamento de processos, o gerenciamento de memória e o sistema de arquivos – em suma, irá ser elucidado como operam os "bastidores" dos aparelhos com esse SO.

Para que se possa realizar tal discussão, estaremos fundamentados em uma série de pesquisas em links e livros que explicam o funcionamento geral dos SOs. Consoante a tais fatores partiremos para o texto propriamente dito.

Palavras-chave: Sistemas Operacionais, Android, Gerenciamento.

Sumário:

1- Introdução:

1.1- O que é Android?

Figura 1 – Mascote Android

O Android é um sistema operacional desenvolvido pela empresa Google para dispositivos móveis baseado no Linux (celulares Tablets e afins) . O Sistema é responsável por gerenciar todas as tarefas do seu aparelho, além de fornecer uma interface visual para que seja possível sua interação com usuário.

Um sistema operacional é um conjunto de programas que gerenciam todas as tarefa e organização de um dispositivo, e nos fornece uma interface visual para que possamos interagir com um sistema eletrônico sem necessariamente saber o que acontece dentro dele.

São sistemas operacionais, ou "S.O.", Windows, Mac OS ou o Ubuntu, que permitem que qualquer pessoa consiga, de forma intuitiva, utilizar praticamente todos os recursos de um computador, ainda que o usuário não saiba escrever uma linha sequer de programação.

O mesmo vale para o Android, mas esse foi desenvolvido para aparelhos moveis, inclusive hoje é sem dúvidas o mais usado , e tem compatibilidade com praticamente todos os dispositivos moveis existentes, inclusive os da principal concorrente Apple com o IOS.

2- História do Sistema Android:

O Android, atual sistema operacional do Google, foi desenvolvido em 2003 por Andy Rubin, Rich Miner, Nick Sears e Chris White; a elaboração do SO foi feita inicialmente com essência de ser um sistema móvel capaz de se adaptar as preferências e localidades do seu dono (em suma, o Android defende em qualquer ponto a personalização de sua interface pelo usuário). A interface foi desenvolvida com um tom de simpleza, sendo, portanto, funcional e com alguns instrumentos integrados, com isso ofereciam ao mercado um SO para customização dos desenvolvedores.

Após dois anos, o Android Inc foi comprado pela Google, nascendo assim a Google Mobile Division. Mesmo com a aquisição da empresa Android, o Google propriamente dito não tinha grande confiança em sua aquisição, dado que a empresa ainda não tinha “mostrado para o quê veio”. Compreendendo essa realidade, a Google criou um “pequeno” concurso (com 10 milhões me prêmios) para que os desenvolvedores criassem aplicativos para a marca . Consoante a isso o 1° celular da Android foi criado (Figura 2).

Figura 2 – Android Sooner.

É válido citar que o modelo do Android Sooner teve que ser praticamente abandonado, dado que o iPhone havia sido revelado e mudou todo o mercado. Consoante a essa mudança no mercado e sob o avanço de mais dois anos, à 2007, ocorreu uma união em um consorcio dos fabricantes Samsung, Sony, HTC, T-Mobile, Nextel e o próprio Google; o objetivo de tal encontro era a criação de uma plataforma de código aberto, o resultado foi a criação do primeiro Android comercial. (Figura 3)

Figura 3 - Smartphone HTC T-Mobile G lançado em 2008.

Esse último aparelho ainda assim não gerou grande confiança por parte do público, dado que o Android não tinha grande reconhecimento; tal realidade fez com que outros produtores de Smartphones subjugarem a empresa afirmando que eles não iriam ter um grande sucesso ou que não eram fortes competidores. Tal realidade evoluiu de forma que o Android se tornou a marca mais usada no mundo, e empresas como a Nokia ou a Microsoft chegam atualmente a ter 2,7% do mercado de telefones com marcas usando seus Sos. O Android foi evoluindo e em 2014 chegou a ter 81,5% do mercado com o seu SO.

2.1- Modelos para o Sistema Operacional:

Desde a sua criação, em 2003, o Android teve uma série de 11 modelos criados, cada um tendo suas próprias considerações e peculiaridades. Inicialmente, é válido indicar ao leitor que nesse ponto do trabalho, não ocorrerá grandes especificações dos sistemas operacionais, esse é somente um trecho complementar a história do sistema operacional. Deixando isso claro, partimos para os sistemas já criados:

Paralelamente ao lançamento do Android Sooner, indicado na figura 1, foi criado a primeira versão do Android, o Alfa 1 (logo indicado na figura 4). Esse SO por ser o primeiro lançado pela Google, possuía um número considerável de aplicações para a sua é poca, as mais significativas advindas com o Alfa 1 foram a concepção da barra de notificações, a conexão com aplicativos da Google(tais como o Gmail, o GoogleMaps *et coetera*) e o início do SO aberto.

Figura 4 _ Logo do Android Alfa 1

Avançando um pouco no tempo, chegamos ao Android Beta(versão 1.1), o qual foi elaborado para o HTC T-Mobile, indicado na figura 2. Essa versão mais atualizada do SO já permitia a atualização dos aplicativos por via do app Market (futura playstory), possui serviço para câmera e também serviços melhores de sincronização com o Google.

No ano seguinte após o lançamento do Android Beta, iniciou-se a "época dos doces" para o SO, a partir desse ponto cada novo modelo proposto recebera o nome de uma guloseima. Consoante a isso, temos o lançamento do Android Cupcake esse padrão também começou o touchscream. Em setembro desse mesmo ano, temos também o lançamento do Android Donut, porém ambos os SOs forma rapidamente substituídos.

Consoante a "queda" dos últimos SOs, nasceu o Android Eclair e com ele temos a mudança no sistema de navegação do Google Maps, além do reconhecimento de voz para as mensagens de texto.

No ano de 2010 surgiu o Froyo (a versão 2.2), o qual levou os recursos de voz a um novo patamar. Ao final do ano de 2010 também foi lançado o Android 2.3 – Gingerbread, o qual possuía suporte a tecnologia NFC e chamadas por via da internet. Mais um ano e alcançamos o modelo Honeycomb, o modelo que trazia suporte aos tablets juntamente a ele.

Com essa última produção da Google não ocorreram alterações/ganhos expressivos por parte dos SOs criados, somente "melhorias" nas versões posteriores de forma que nesse ponto é válido mais uma noção honrosa aos Androids criados, no caso: o Ice Cream Sandwich, o Jelly Bean, o Kit Kat(diga-se de passagem, o foco deste trabalho), o Lollipop, o Marshmallow, o Nougat, o Oreo, o Pio e por fim o Android 10.

3- Gerenciamento de Processos Android:

Na maioria dos casos, uma aplicação Android é executada em um processo próprio. Esse processo é criado para a aplicação pelo sistema operacional quando o código da aplicação deve ser executado, e permanece sendo executado até que a aplicação não o necessite mais e também que o sistema operacional tenha redeclarado a memória que havia sido entregue à aplicação e ao processo.

Embora não seja comum, no Android o ciclo de vida de uma aplicação não é controlado diretamente por ela mesma. O sistema operacional tem controle sobre isto, e leva em consideração vários aspectos do ponto de vista do que é melhor para o usuário. Isso significa que o sistema operacional tem total liberdade para terminar de maneira forçada uma aplicação se isso for necessário para manter o sistema funcionando adequadamente.

Quando o dispositivo estiver com pouca memória disponível, o Android classifica todos os processos por uma hierarquia de importância, baseado nos componentes que estão sendo executados em cada processo e o estado desses componentes.

Figura 5 _ Mascote Android com ferramenta

Quatro tipos de processos são distinguidos:

3.1) Foreground:

É um processo que é necessário para o que o usuário está atualmente fazendo. Muitos componentes diferentes podem causar um processo ser classificado como primeiro plano. Um processo é classificado como estando em primeiro plano caso qualquer uma das seguintes condições seja satisfeita:

- Está sendo executada uma "Activity" com a qual o usuário está interagindo ("onResume()" foi chamado).

- Um "BroadcastReceiver" está sendo executado ("onReceive()" foi chamado).

- Tem um "Service" que está sendo executado em um de seus "callbacks" ("onCreate()", "onStart()", ou "onDestroy()" foi chamado).

Só podem existir alguns processos desse tipo no sistema ao mesmo tempo, e estes só serão terminados a força em último caso, ou seja, se o dispositivo estiver com tão pouca memória que nem mesmo estes processos essenciais possam ser executados. Geralmente, se chegar nesse ponto, é necessário que isso seja feito para manter a responsividade da interface do usuário.

3.2) Visible:

É um processo que está executando trabalho que o usuário tem conhecimento sobre, ou seja, o ato de o sistema operacional terminar esse processo, para salvar memória por exemplo, seria certamente percebido pelo usuário e o impactaria de maneira negativa. Um processo é considerado visível nas seguintes condições:

- Está executando uma "Activity" que é visível para o usuário no "screen" mas não no "foreground" ("onPause()" foi chamado).

- Tem um "Service" que está sendo executado como "foreground service" ("startForceground()" foi chamado).

- Está hospedando um "Service" que o sistema está usando para alguma coisa que o usuário possui conhecimento sobre.

O número de processos desse tipo no Android é menos limitado em relação aos processos "foreground", mas ainda assim é controlado. Esse tipo de processo ainda é considerado extremamente importante e só será terminado de maneira forçada em último caso, assim como os processos de "foreground".

3.3) Service:

É um processo que está hospedando um "Service" ("startService()" foi chamado). Embora esses processos geralmente não são vistos pelo usuário, eles ainda costumam estar fazendo trabalho que o usuário tem conhecimento sobre, ou seja, terminar esse tipo de processo resultaria em um impacto negativo na experiência do usuário. Por esse motivo, esses serviços também são considerados importantes e só serão finalizados de maneira forçada se isso for necessário para manter os processos de maior importância funcionando.

Processos que possuem mais de 30 minutos de duração podem ter sua importância reduzida, para que outros processos possam ter chance de ser executados, isso previne que um processo acabe monopolizando os recursos do dispositivo impossibilitando que o sistema operacional mantenha a experiência do usuário fluida.

3.4) Cached:

É um processo que não precisa mais ser executado, então o sistema pode terminá-lo assim que desejar, e liberar os recursos para serem utilizados por outro processo. Em um sistema funcionando corretamente, estes serão os únicos processos envolvidos em manutenção de recursos: um sistema em bom funcionamento terá múltiplos "cached processes" sempre disponíveis e regularmente terminar os antigos conforme necessário. Somente em casos críticos e não desejáveis que o sistema estará em uma situação onde todos os "cached processes" foram terminados e será necessário partir para terminar outros tipos de processos.

Esses processos usualmente possuem instâncias de "Activity" que não são atualmente visíveis para o usuário ("onStop()" foi chamado). Tais processos são guardados em uma lista, a ordenação dessa lista é dependente de implementação, mas geralmente ela guarda processos mais importantes primeiro.

3.5 - Comunicação entre processos:

O Binder é usado para comunicação entre processos no sistema Android. Um módulo no kernel é implementado em "drivers/misc/binder.c" para isso. Toda a comunicação feita entre processos no sistema Android passa pelo binder. Essa é uma operação interna que o programador não precisa se preocupar pois é abstraída pelas bibliotecas do sistema.

3.6- Escalonamento de processos:

O Android, a exemplo do Linux, divide processos em três categorias:

- Processos Interativos

- Processos Batch

- Processos Tempo Real

Em cada categoria os processos são posteriormente divididos em I/O Bound ou CPU Bound. O Escalonador do Android não faz distinção entre processos batch e iterativos, somente diferencia os processos em tempo real. O escalonamento do Android é do tipo preemptivo. O algoritmo divide o quantum em epochs. Os processos quando são criados recebem um quantum no início de uma epoch. Desse modo, o valor de quantum não é fixo, e depende de cada processo.

3.7-Threads:

Processos podem conter múltiplos threads, como é comum em sistemas operacionais baseados no Linux. A maior parte das aplicações Android consistem de múltiplos threads para separar input de interface de usuário (UI input) de operações de entrada e saída (I/O operations) ou cálculos demorados. Os threads usados em uma aplicação Android são threads do Java.

O Android runtime permite que cada thread execute uma instância própria da máquina virtual. Mesmo a linguagem usada sendo Java, a máquina virtual não é a máquina padrão Java, mas sim a máquina virtual Dalvik, que é otimizada para dispositivos móveis. Essa máquina virtual foi desenvolvida especialmente pela Google para melhorar o desempenho do sistema Android nos dispositivos móveis.

3.8- Applications & Tasks:

Aplicações Android são executadas por processos e incluem threads. Os termos tasks e aplicações estão interligados fortemente, visto que uma task pode ser vista como uma aplicação para o usuário. Na verdade, tasks são uma série de atividades de possivelmente múltiplas aplicações. Tasks basicamente são um histórico lógico das ações do usuário, por exemplo, o usuário abre uma aplicação de email onde então ele abre um email específico com um link que abre o navegador. Nesse cenário a tarefa incluiria as duas aplicações (e-mail e navegador). Uma vantagem do conceito de task é a oportunidade de permitir ao usuário retroceder suas ações como regressar em uma pilha.

3.9- Application internas:

A estrutura interna de uma aplicação Android é baseada em quatro diferentes componentes, que são: Activity, Service, BroadcastReceiver e ContentProvider. Uma aplicação não consiste necessariamente dos quatro componentes, mas para representar uma interface gráfica precisa haver pelo menos uma Activity.

Aplicações podem abrir outras aplicações ou componentes específicos de outras aplicações mandando um Intent. “Intents“ são formados pelo nome da ação desejada, além de outras coisas. O “IntentManager” resolve os pedidos de “intent” e inicia as devidas aplicações ou componentes.

Services e Broadcast Receivers permitem que aplicações executem tarefas em segundo plano e providenciem funcionalidades para outros componentes. Broadcast Receivers podem ser disparados por eventos e são executados apenas por um curto período de tempo, já um serviço pode ser executado por um longo tempo.

O código compilado de uma aplicação e dos recursos adicionais como bibliotecas, imagens e outros arquivos necessários são compactados em um único arquivo .apk que forma o executável da aplicação Android.

4- Gerenciamento de Memória Android:

4.1- Visão geral do gerenciamento de memória:

Os sistemas atuais e principais Android (as versões de mercado atual [Android Runtime (ART) e a máquina virtual Dalvik]) se utilizam de paginação e mapeamento em memória (mmapping) para gerenciar a memória.

4.2- Memórias do Android:

No sistema Android possuímos o armazenamento Android e a memória Android, assim como as tarefas Android.

Basicamente qualquer memória que um app modifique, seja alocando novos objetos ou tocando em páginas mapeadas em memória. Continuará a residir um espaço físico na RAM e não pode ser retirada até a sua "despaginação".

A principal e muitas vezes única maneira de liberar a memória de um app, se constitui em as referências de objetos mantidas por ele, disponibilizando a memória para o coletor de lixo.

A exceção desse que foge a esse "despaginação" principal, são todos os arquivos mapeados em memória sem modificação, como código, podem ser despaginados da RAM caso o sistema queira usar essa memória em outro lugar.

A memória RAM de um celular atual pode variar de 2 até 8 GB de RAM. Só o sistema operacional pode usar uma porção considerável de RAM. Assim sendo, você nunca consegue usar a RAM na sua totalidade.

Uma das razões mais comuns para lentidão dos aparelhos Android não se dá por causa do processador, mas sim pela falta de memória RAM, disponível para uso.

A plataforma Google Android tem o hábito de manter processos correndo no fundo - mesmo estando inativos - e assim consomem muita memória RAM , por isso uma boa prática de usuário seria fechar apps de segundo plano que não estão sendo utilizados.

4.3- Partilha de memória:

Para ajustar tudo o que é necessário na RAM, o Android tenta compartilhar páginas da RAM entre os processos. Isso pode ser feito das seguintes maneiras:

Por Alocação Particionada Estática conhecida como método do Zigoto, em que cada processo do app é bifurcado a partir de um processo existente denominado "Zigoto". Que tem seu início junto com a inicialização do sistema e carrega códigos e recursos comuns do framework, como temas da atividade.

Para cada novo processo do app, o sistema bifurca o processo Zigoto , logo depois carrega e executa o código do app no novo processo.

Por Alocação Particionada Estática de mapeamento , em que um conjunto de dados estáticos é mapeada em memória para um processo. Essa técnica permite que os dados sejam compartilhados entre processos e paginados quando necessário.”

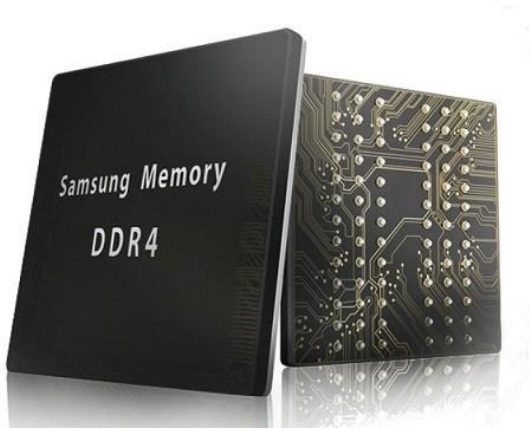

Figura 6 _ Memoria DDR4 Samsung

4.4- Coleta de lixo:

Quando uma parte da memória é destinada a uma aplicação, e esta memória não é utilizada pelo programa, ela é liberada de volta para o heap (gerenciamento de memória virtual) , por meio do sistema operacional que fica em verificação para a coleta de lixo.

A coleta de lixo tem dois objetivos:

-Localizar objetos de dados em um programa que não pode ser acessado no futuro.

-Recuperar os recursos usados por esses objetos.

"O heap de memória do Android é geracional, o que significa que há intervalos distintos de alocações rastreadas com base na vida útil esperada e no tamanho de um objeto alocado."

Ou seja , a parte de alocação de um objeto é baseada no período que ele ainda pode ocupar a memória, podendo receber uma geração 'jovem' ou mais antiga , e com isso pode receber uma determinada prioridade na lista de memória.

"Cada geração de heap tem o próprio limite máximo do volume de memória que os objetos podem ocupar. Sempre que uma geração começa a ser preenchida, o sistema executa a coleta de lixo para tentar liberar a memória.

A duração da coleta de lixo depende da geração de objetos que ela está coletando e da quantidade de objetos ativos em cada geração."

A coleta de lixo é baseada em um conjunto de critérios do próprio sistema, que determinam qual o momento para realizar uma coleta de lixo. A coleta começa quando os critérios do sistema são atendidos pela aplicação, que fica com a execução interrompida para haver a coleta.

4.5- Priorização de deslocação de memória:

Os processos no Android possuem estado e a deslocação de recursos como mostrado anteriormente em processos, porém as memórias desses processos seguem a mesma ordem , a qual é :

- Empty Process — Processos e aplicações que já foram finalizadas, porém mantidos em memória cache para inicialização mais rápida e uso futuro .

- Background process — Processos ou Activities que não estão sendo vistas nem recebendo interação do usuário e não possuem serviços em execução. Mas ocupam memória ativa , mesmo que seja abstraída do entendimento do usuário.

- Started Service Process — Hosting services, sem interação ou visualização do usuário, é aqui que ocorre a partição de memória, por "Zigotamento".

- Visible Process — Processo/aplicação que está visível porém não possui interação. O visor da tela por exemplo, também ocupa uma memória destinada

como uma aplicação, porém pertencente a memória destinada ao sistema operacional.

- Active Process — Processo/aplicação que está sendo usada no momento, a tela está sendo mostrada e o usuário está interagindo, e a memória está sendo alocada conforme a interação do usuário.

4.6- Activity Stack:

Assim como os processos as Activities possuem estados e são gerenciados pelo sistema na realocação de recursos e de uso da memória correspondentes. Os estados de uma Activity são 4, Inativo, Parado, Pausado e Ativo.

Além dos estados, a ordem de uma Activity na Activity Stack também é levada em consideração na deslocação de memória.

Vale lembrar que a Activity Stack é uma pilha LIFO (Last In First Out), ou seja a deslocamento pode ocorrer nessa ordem também.

4.7- Alocar e recuperar memória do app:

No heap , ocorre uma memória com Alocação Particionada Dinâmica cujo, cada processo do app é restrito a um único intervalo de memória virtual. Isso define o tamanho lógico da memória virtual que pode aumentar conforme o necessário, mas apenas até um limite definido pelo sistema para cada app.

O tamanho lógico da memória virtual (HEAP) não é o mesmo que a quantidade de memória física usada , na verdade é menor do que ele disponibiliza para o app.

Ao inspecionar a pilha do app, o Android calcula um valor chamado PSS (Tamanho do conjunto proporcional, em português), que analisa e classifica as páginas limpas e sujas que são compartilhadas com outros processos, em uma quantidade proporcional ao número de apps compartilhados pela RAM. Assim ele faz um Tabelamento de páginas para o app que começou a execução.

O Android só pode reduzir o tamanho lógico do seu gerenciador de memória (heap) quando há espaço não utilizado no final do heap. No entanto, o sistema ainda pode reduzir a memória física usada pelo heap.

No entanto, a recuperação de memória de pequenas alocações pode ser muito menos eficiente uma vez que para realoca-las ocorre um custo de

processamento e isso pode reduzir a eficiência do Sistema durante o realojamento dessas memórias. Fora o problema de alocação que ainda pode ser compartilhada com algo que ainda não foi liberado.

4.8- Restringir memória do app:

Para manter um ambiente multitarefa funcional, o Android define um limite rígido para o tamanho de (memória virtual) heap de cada app. O limite exato é difícil de prever, mas o tamanho da memória virtual concedida, varia entre os dispositivos com base na quantidade de RAM disponível para o dispositivo.

Se o app atingir a capacidade da sua memória virtual designada inicialmente o heap pode tentar alocar mais memória, podendo receber um Out_Of_Memory_Error(Erro de falta de memória) .

Em alguns casos, o sistema pode ser consultado para determinar exatamente quanto espaço de memória virtual está disponível no dispositivo atual. Por exemplo, para determinar quantos dados devem ser mantidos em cache.

4.9- Alternar entre apps:

Como um sistema multitarefa, os usuários podem alterar entre apps. Com isso o Android mantém em um cache os apps que não estão em primeiro plano, ou seja, não estão visíveis para o usuário ou estão executando serviços de primeiro plano, como reprodução de músicas.

Para esses casos ,quando o usuário abre um app pela primeira vez, um processo é criado para ele. Mas, quando o usuário sai do app, esse processo não é encerrado e ele continua sendo executado pelo sistema. Por meio do cache. Se o usuário retornar ao app mais tarde, o sistema utilizará o processo, tornando a alternância de apps mais rápida.

Caso o sistema fique sem recursos, como memória, ele encerra processos no cache, isso só acontece caso o programa de primeiro plano requeira muitos recursos do sistema, assim o próprio sistema encerra os processos e cache para liberação de recursos .

5 - Sistemas de Arquivos Android:

O Armazenamento Android é o armazenamento de dados em que se guarda e se gerencia os arquivos gerados pelos programas e pelo usuário.

No Android temos dois sistemas de arquivos, **EXT4** e **F2FS**:

Figura 7 _ Mascote Android com Cartão de memória

5.1- EXT4 :

Quarto sistema de arquivos estendido, ou em inglês “fourth extended filesystem”, é um sistema de arquivos transacional que é capaz de criar até 64.000 subdiretórios.

Possui uma desfragmentação que individualiza cada arquivo e o desfragmenta, assim não necessitando desmontar o disco para desfragmentar o sistema de arquivos inteiro. Este sistema de arquivos é o usado pela maioria dos dispositivos Android conta com diversos recursos como:

-Pré alocação: O sistema de arquivos ext4 permite pré-alocação de espaço em disco para um arquivo.

-Alocação multibloco: O alocador multiblock é usado quando a atribuição atrasada é ativada por um sistema de arquivos, ou quando os arquivos são abertos no modo O_DIRECT.

-Alocação tardia: Ext4 usa uma técnica de execução do sistema conhecida como a atribuição de atraso. Isso melhora o desempenho e reduz a fragmentação, melhorando a alocação de blocos de decisões com base no tamanho do arquivo.

-Suporte para tamanhos maiores de volumes e arquivos: O sistema de arquivos ext4 pode suportar volumes com tamanho até 1 exabyte e arquivos com tamanho até 16 terabytes, mas basicamente nenhuma máquina que roda o sistema Android vai trabalhar com essa quantidade .

-Extensões: Uma extensão é um conjunto de blocos contíguos físico, melhorando o desempenho de muitos arquivos e reduzindo a fragmentação.

5.2- F2FS:

"Flash-Friendly File System". Trata-se do sistema de arquivos criado pela Samsung para o núcleo Linux (em que se baseia o Android). Ele foi criado de forma específica para o armazenamento principal em Flash, ou seja, a forma de memória da grande maioria dos smartphones. SD, em ambos os casos com tecnologia flash. O F2FS foi projetado com base em uma abordagem de sistema de arquivos estruturado em log ,a qual se adaptada a novas formas de armazenamento

Ele suporta vários parâmetros não apenas para configurar o layout no disco, mas também para selecionar algoritmos de alocação e de limpeza , além de conseguir utilizar recursos fortes e muito eficientes como:

-Recuperação em roll-back e roll-forward ;

-Tabela de hash de vários níveis para entradas de diretórios;

-Separação de dados a quente ea frio ;

-Desfragmentação de sistema de arquivos online/desfragmentação de arquivos:

-Criptografia em nível de sistema de arquivos ;

-Descarga de dados periódica interna.

5.3- Categorias de locais de armazenamento:

O Android oferece dois tipos de locais de armazenamento físico para guardar todos sos arquivos do usuário e os arquivos do sistema em si :

O armazenamento interno e o armazenamento externo que geralmente se apresenta e forma de cartões de memória SD .

Enquanto o armazenamento interno está sempre disponível em todos os dispositivos, tornando-se um lugar mais confiável para colocar dados indispensáveis para seu app, as unidades removíveis, como um cartão SD, aparecem no sistema de arquivos como parte do armazenamento externo que nem sempre são totalmente confiáveis para se depositar dados chave de aplicações .

5.4- Acesso ao diretório raiz (OTG) e cartão SD:

Acesso de gravação aos diretórios de armazenamento interno, Este acesso de gravação inclui acesso direto ao caminho do arquivo, Os aplicativos que recebem essa permissão não concede acesso aos diretórios específicos do aplicativo que pertencem a outros aplicativos (como um arquivo compartilhado).

“*Quando um aplicativo tem a permissão MANAGE_EXTERNAL_STORAGE, ele pode acessar esses arquivos e diretórios adicionais usando a API MediaStore ou caminhos de arquivo diretos*.”

6- Curiosidades:

6.1- Como surgiu o robô?

O mascote do Sistema Operacional foi criado pela designer Irina Blok. A inspiração veio dos desenhos que ficam na porta dos banheiros distinguindo o acesso entre o masculino e o feminino. Internamente, o robô é chamado de Bugdroid.

Figura 8 _ Mascote Android – Bugdroid

6.2- As versões têm nomes de doces

Uma outra curiosidade é que Todas as versões do Android têm nomes de doces, seguindo uma sequência alfabética.
A tradição começou no Android 1.5. e assim seguiu.
São eles: Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo e Pie.

Segundo o porta-voz do Google, os nomes surgiram como uma piada interna na empresa, começando com o Android 1.5 Cupcake. Em 2013, a empresa explicou que esta tradição ocorre porque os smartphones e tablets com seu sistema adoçam a vida dos usuários.

Figura 9 _ Mascote Android com doces de sua respectiva versão

7- Conclusão:

Com tudo o que foi relacionado acima, podemos afirmar que o sistema operacional Android é um sistema completo. Com toda certeza um dos mais bem colocados do mercado, tanto pela sua fácil adaptação com todos os dispositivos móveis uma vez que seu Gerenciamento de processos e gerenciamento de memórias não dependem totalmente de especificações de hardware para terem um desempenho elevado.

Sua relação com o usuário também é facilitada pelo seus gerenciadores de e escalonadores de processos, cujo os quais bebem da fonte Linux, mas principalmente por ter algumas da aplicação mais interativa da própria Google, que já torna o ambiente mais fácil e mais amistoso para o usuário.

Com seus sistemas de Threads e seu sistema de coleta de lixo ele consegue aproveitar bem o desempenho do hardware que os dispositivos tem a oferecer. Também consegue por meio do seu sistema de Cache manter aplicações rodando fora da tela e mesmo assim sendo executadas de forma que o aparelho não seja sobrecarregado e tenha o melhor dos desempenhos a oferecer.

Os avanços no passar do tempo tanto na tecnologia dos dispositivos , quanto nas versões de compatibilidade Android são notáveis, mas o que mais é interessante ressaltar do sistema, é que mesmo com tantas distinções de recursos de um aparelho para o outro o mesmo sistema é redado e entrega o melhor que aquele aparelho tem a oferecer de maneira inteligente e organizada.

Seus recursos de controle das aplicações e regulação de memória assim como suas formas de sistema de arquivos demonstram que o Android não só é capaz de ser a melhor plataforma de adequação aos mais distintos dispositivos, mas também um dos mais eficientes em quesito aproveitamento do Hardware em relação a experiencia do usuário.

A abstração doque se passa por dentro do aparelho e por meio do sistema operacional é tão bem executada que os desenvolvedores não tem que se preocupar muito com o aparelho , apenas com algumas especificações do aplicativo, podendo restringir os recursos ou pedir mais. Mas eles sabem que a aplicação será bem efetuada independente de quem a use e em qual dispositivo (se compatível) esteja usando.

8- Referências Bibliográficas:

1. https://www.nextpit.com.br/como-gerenciar-memoria-ram-Android
2. Flash Memory Filesystem, Korea Linux Forum
3. A New File System Designed for Flash Storage in Mobile, Embedded Linux Conference Europe
4. https://www.nextpit.com.br/sistemas-de-arquivos-Android-f2fs-vs-ext4
5. https://developer.Android.com/topic/performance/memory-overview?hl=pt-br
6. https://cucha.com.br/a-plataforma-Android-e-o-gerenciamento-de-memória-424a8b25ddb
7. https://elias.praciano.com/2013/03/gerenciamento-de-memoria-no-Android/
8. https://developer.Android.com/guide/components/activities/process-lifecycle
9. Analysis of the Android Architecture - Studienarbeit von - Stefan Brahler
10. Artigo: Anais do Congresso de Sistemas Operacionais do CPoli da UCPEL, VOL. 1, NO. 1, 2016 1

11. https://developer.android.com/?hl=pt-br

12. https://developer.android.com/training/data-storage/manage-all-files?hl=pt-br

13. https://www.techtudo.com.br/artigos/noticia/2011/01/afinal-o-que-e-android.html

14. https://www.android.com/intl/pt-BR_br/

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

Trabalho Sistemas Operacionais

Docente: Diego Ferreira dos Santos

Discentes:

Rodrigo Luis Tavano Bosso – PC3005623

Rafael Abreu Coelho - PC3004481

Danylo Expedito de Sabino _ PC3005674

Sistema Operacional Android

FIM