Tornando a árvore de diretórios dinâmica com UnionFS, FUSE e Inotify

Hisham H. Muhammad
André Detsch

gobolinux.org

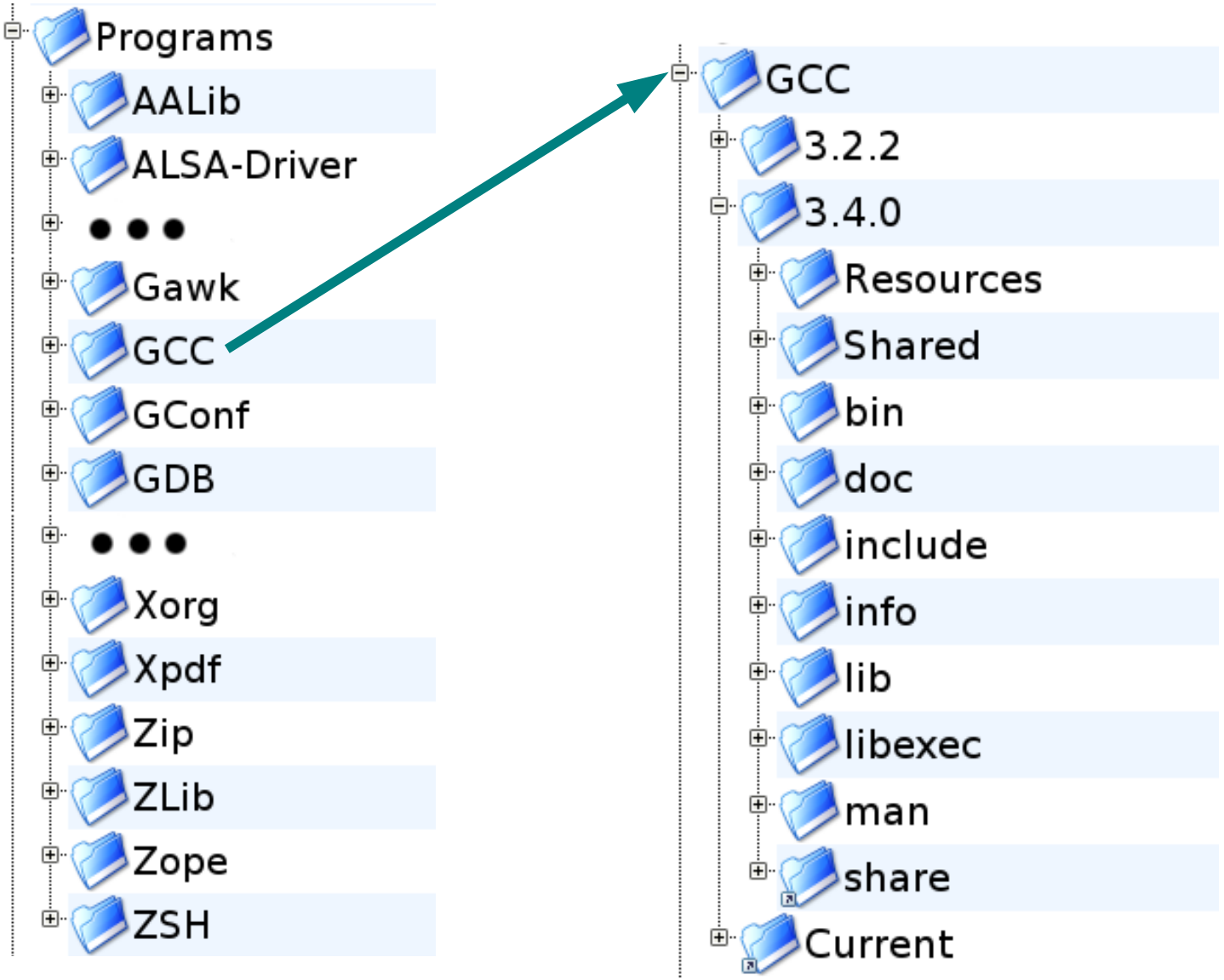
Resumo

- O que é o GoboLinux?
- Presente
  - Evoluções já realizadas
  - Dificuldades de flexibilização nas aplicações
- Futuro
  - UnionFS, FUSE e Inotify como ferramentas para resolver os problemas de flexibilização existentes
  - Alterações previstas para o GoboLinux
- Considerações finais

O que é o GoboLinux?

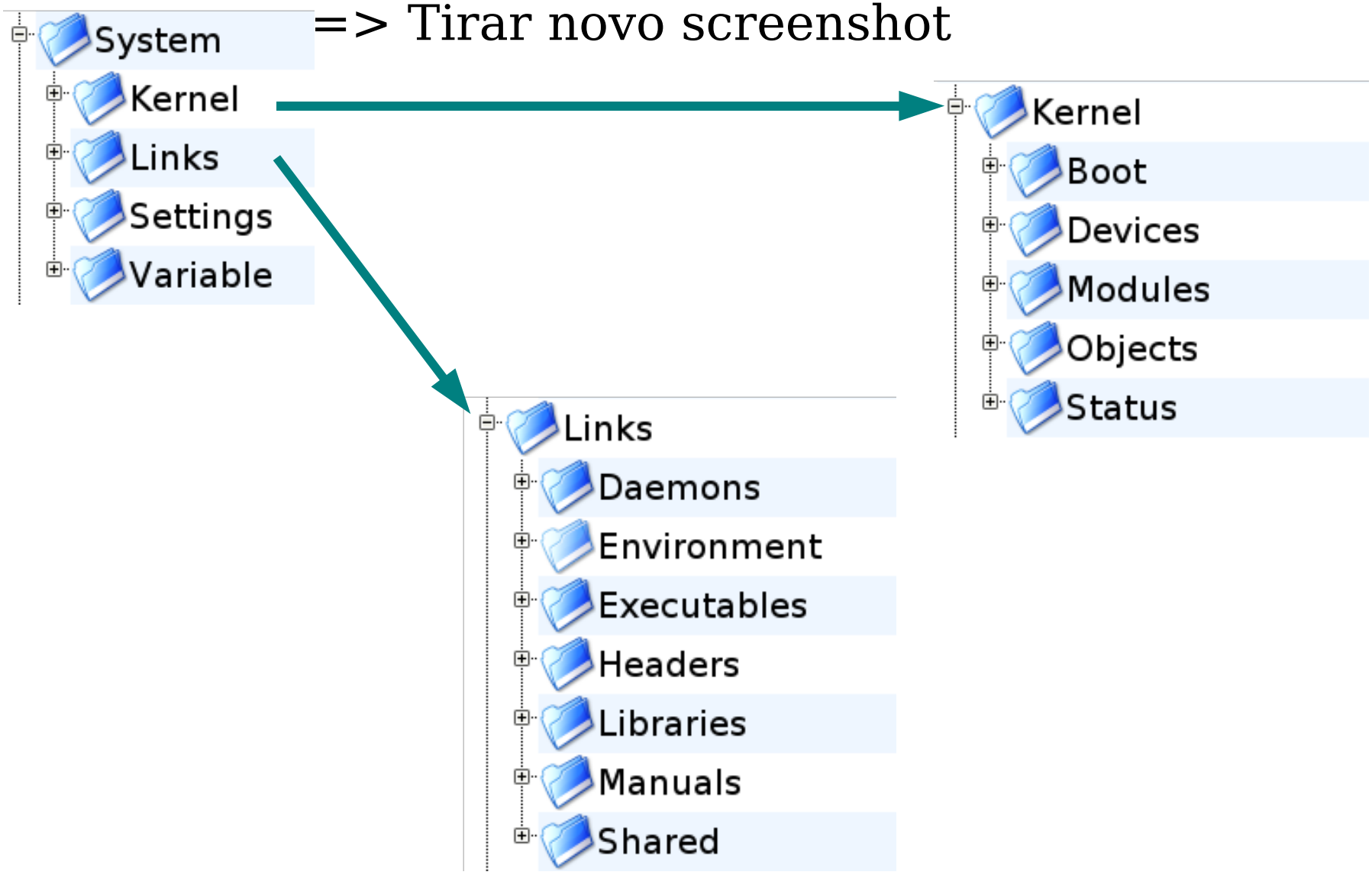
- Distribuição realmente alternativa
- Nova organização da estrutura de diretórios, adaptada aos padrões de uso atuais
- Alguns marcos
  - 2002, criação
  - 2003, kuro5hin, slashdot
  - 2004, Revista do Linux
  - 2005, cooperação com IBM

Estrutura de diretórios



Estrutura de diretórios

Trocar o Daemons para Tasks
=> Tirar novo screenshot



Características gerais

- Links de compatibilidade (/bin, /usr, /etc...), que podem ser ocultados -> GoboHide
- Pacote binário é um “tar.bz2”
- Remoção é um “rm -rf”
- Manutenção garantida: independência de pacotes, ou gerenciador de pacotes
- Scripts automatizam tarefas, mas são opcionais

Evolução da árvore Gobo

Principais alterações

- /System/Kernel/
  - Boot : questão da partição separada (/System/Boot -> /System/Kernel/Boot)
  - Modules: /System/Links/Libraries/modules (/usr/X11R6/lib/modules do XFree86)
  - Devices, .../Status, .../Objects: /dev, /proc e /sys -> GoboHide
- /System/Links/Daemons virou /System/Links/Tasks

Arquivos não gerenciados

- /System/Variable
  - lock, run, log
  - não se quer apagar os logs após desinstalar um servidor web
- Resources/Unmanaged (novo na 013)

Arquivos compartilhados

- /System/Links/Shared adicionado depois...
  - Esquema do link invertido
  - ...
  -

Problema geral: programas que assumem um caminho compartilhado

Abordagens atuais

- Variáveis de ambiente
  - GTK_PATH e amigos
  - solução proposta por algumas aplicações
    - apontam para /System/Links
  - nem sempre seguida
  - "pouco escalável"
    - pode ser problemático automatizar para diferentes shells

Abordagens atuais

- /usr/lib/python2.3 e amigos
  - algumas aplicações "estendem" a árvore de diretórios
  - incluindo seu próprio esquema de versões
  - o que é lib? é apenas para bibliotecas dinâmicas?
  - arquivos independentes de plataforma não deveriam estar em share?
  - ponto de discordância entre distribuições

Por que se importar com isso?

- Questão chave em um sistema modular
- Pense no usuário não-root
  - por isso mesmo soluções como variáveis de ambiente (quando disponíveis) são vistas como "exceção"

Apresentando /System/Index

/System/Index

- A próxima geração da árvore GoboLinux
  - manter a "separação física" dos pacotes
  - princípio "the filesystem is the package manager"
- Oferecer realmente uma visão unificada
  - /System/Index é um índice

Como compilar programas?

- É preciso uma forma _garantida_ de compilar programas em um diretório único e capturar os arquivos para compartimentalizá-los em /Programs
- Historicamente, alternativas existentes eram pouco satisfatórias:
  - Trap de chamadas de sistema (checkinstall, etc): não é 100% seguro
  - make com um prefix, make install com outro: não é 100% suportado

UnionFS

- <O que é...>
- Utilização na compilação
  - UnionSandbox
  - configure --prefix=/System/Index
  - os arquivos serão "escritos" no diretório de índice, mas serão redirecionados via UnionFS.
  - mesmo escritas na árvore Unix 'legacy' serão redirecionadas

Usos do UnionFS

- Sandbox
  - Releases recentes dos Scripts Gobo já permitem usar UnionFS como sandbox (em oposição ao usuário não-privilegiado 'fibo')
- Abstrair a "localização física" dos programas
  - Múltiplas partições (o texto I am not clueless já pregava isso)
- LiveCD
  - Camada ramdisk read-write, boa flexibilidade

Índice...

- Índice -> banco de dados!? WTF?
- Não! É live!
  - sabemos bem dos problemas de sincronia de package managers
- Listener já existe
  - "technology preview" do que é possível
  - Monitorar /Programs utilizando Inotify

A árvore dinâmica

ViewFS

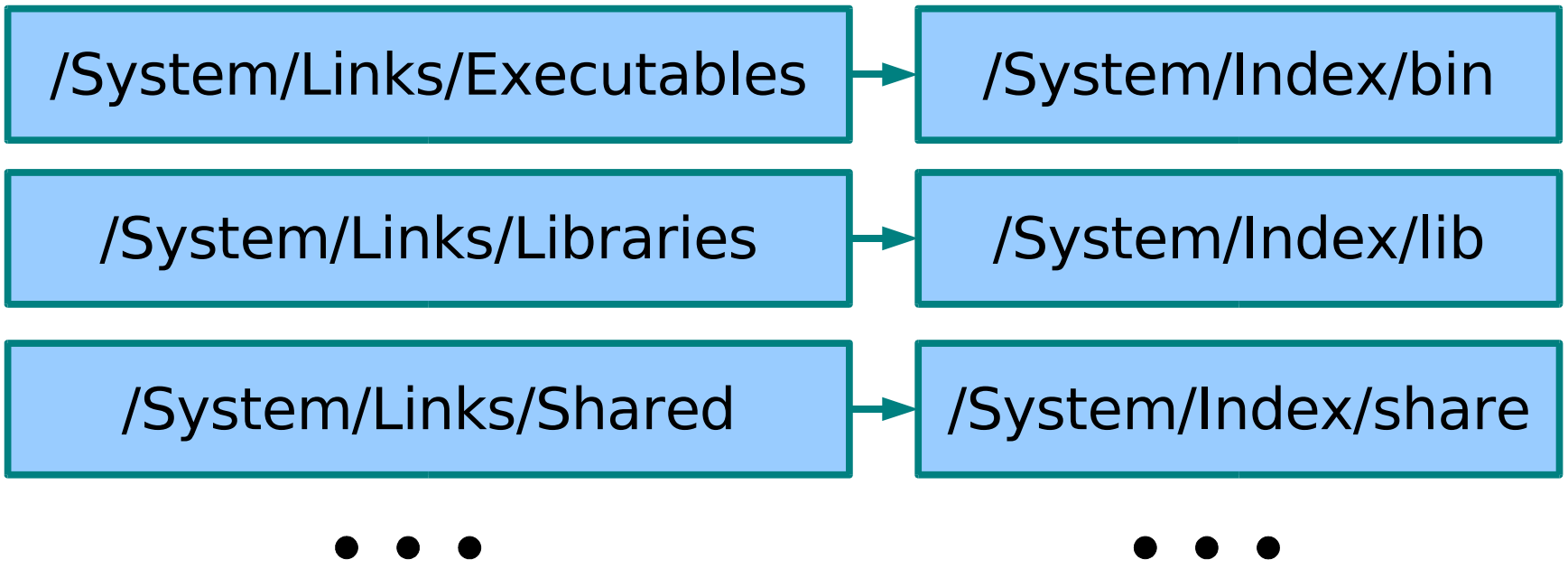
- Sistema de arquivos montado em /System/Index
  - Implementado como um daemon em espaço de usuário, usando FUSE
- Provê visão unificada do sistema
  - Substitui hierarquia /System/Links
- Aparece como uma árvore de links
  - read()s são realizados diretamente no arquivo

Por que “dinâmica”

- Diferentes programas podem “enxergar” índices diferentes
  - Resolve o problema de dependências conflitantes
- Exemplo: dois binários diferentes podem acessar /usr/lib/libc.so.6 diferentes
- Esse caminho passa a ser virtualizado
- Baseado no esquema de dependências
  - Ganhando um facelift: GTK+ [2.0,)
- Inotify integrado no daemon do ViewFS

O que muda

Comparando


/System/Links/Executables
/System/Index/bin
/System/Links/Libraries
/System/Index/lib
/System/Links/Shared
/System/Index/share

Simplificação de conceitos

- Atualmente Gobo precisa tratar cada categoria de arquivos explicitamente
- Não é necessário o esquema de links invertidos para "share"

Nomes mais 'feios'?

- GoboLinux sempre possuiu a noção de "sub-árvores Unix" dentro de sua estrutura de diretórios
- Visão abaixo de /System/Index/... fica consistente com /Programs/Foo/1.0/...

Quebra de compatibilidade?

- GoboLinux sempre primou pela flexibilidade da árvore de diretórios
  - Rootless GoboLinux é um exemplo: /home/hisham/Programs...
- Assim, para nós, o "custo de mudar" é menor
  - Compile, nossa ferramenta de compilação, utiliza definições de caminho de alto nível
- Compatibilidade binária no mundo Linux é constantemente quebrada de qualquer jeito
  - Basta atualizar GCC e Glibc

Conclusões

- Traremos a paz ao mundo
- You heard it first here

"*Que mundo é esse onde nem mesmo diferentes versões de um programa podem coexistir*"

- André Detsch