MacのSandboxとXPC

～ToyViewer改造中

京都産業大学　荻原剛志

KOF 2012 (2012.11.10)

何の話かというと

- 私はNeXTstep時代からのアプリをMac用に手直しして延々と使っている。
  - Mac OS X 10.8に対応するため、これまでのアプリの構成を作り直す必要が。
- Sandboxに代表されるセキュリティ強化
  - 利用者レベルのカスタマイズの余地が小さく
  - アプリケーションからのアクセス制限
  - 下請けプロセスとの通信（XPC）

題材：**ToyViewer**

- 画像ビューア
  - 数種類の画像形式を読み書きできる
  - いくつかのエフェクトを適用できる
  - App Storeで公開中（無料）

Version 5.1.2 (Jan. 2011)

- そもそもなぜ自作？
  - NeXTの時代はJPEGのビューアすら標準ではなかった
  - Web用画像を作るのに必要な程度の機能が中心
  - NeXT / Mac のプログラミングの練習も兼ねて

Ver.0.7 (1995)

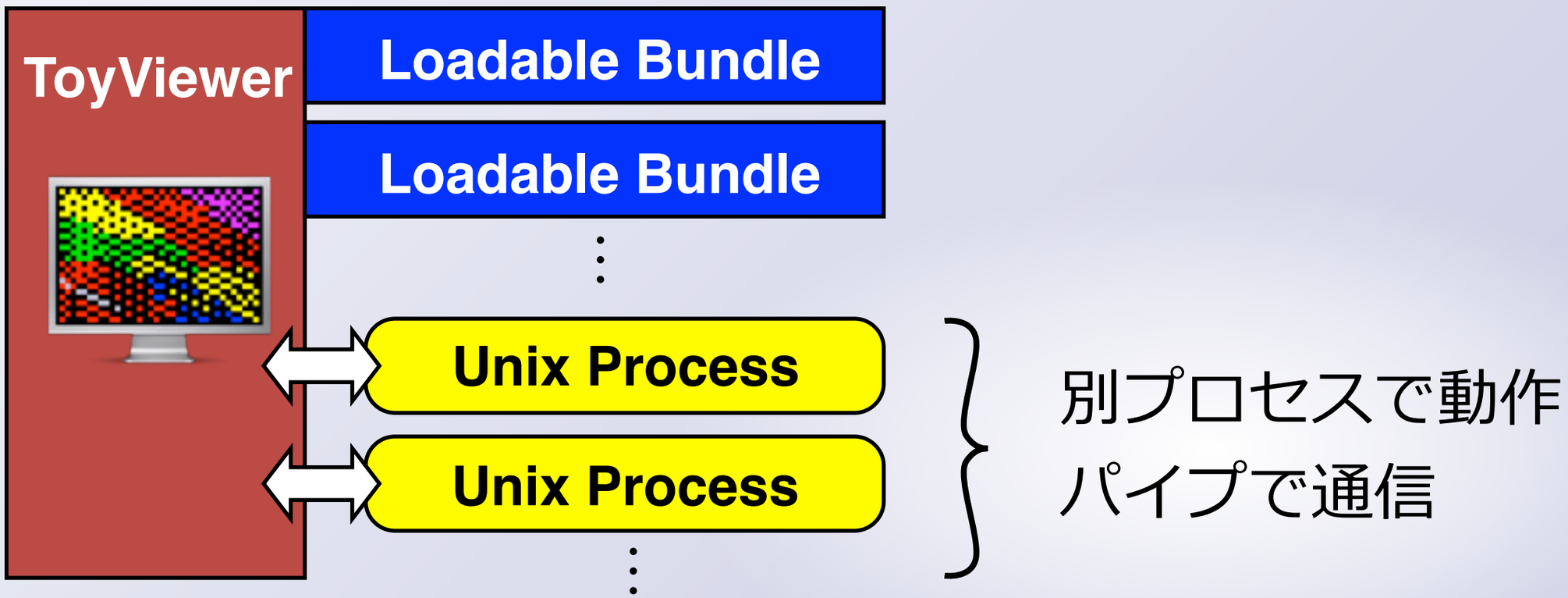
- 世界的に好評を頂いている
  - 日／英／仏／独／中（繁）に対応（日英以外は寄付）
  - Version 4.82 (2007)まではソースも公開
  - GNUstepに移植もされた

ToyViewer の構造

- 画像エフェクトは Loadable Bundle
- 非標準のファイル入出力は別プロセス

別プロセスにすると？

- 入出力で失敗してもアプリ本体はコケない
  - エラー原因で多いのは不正なファイル
- UNIXのコマンドラインツールを簡単に利用可
  - 半分はフリーソフト（PPM形式）



ToyViewer のフィルタ

Version 5.1.2

形式	Read	Write	備考
GIF		**W**	Cocoa APIでは不十分
PNG	**R**	**W**	Cocoa APIでは不十分
BMP	**R**	**W**	Cocoa APIでは不十分
JBIG	**R**	**W**	
JPEG2000	**R**	**W**	Cocoa APIでは不十分
Sun Raster	**R**		
PCX	**R**		
Photo CD	**R**		
MAG	**R**		パソコン通信当時の画像形式
XBM	**R**	**W**	
icns(icon)		**W**	リソースフォーク

App Sandbox

- iOSで成功しているセキュリティ管理を
  Macのアプリケーションにも持ち込みたい

Entitlements

Containers

Kernel Enforcement

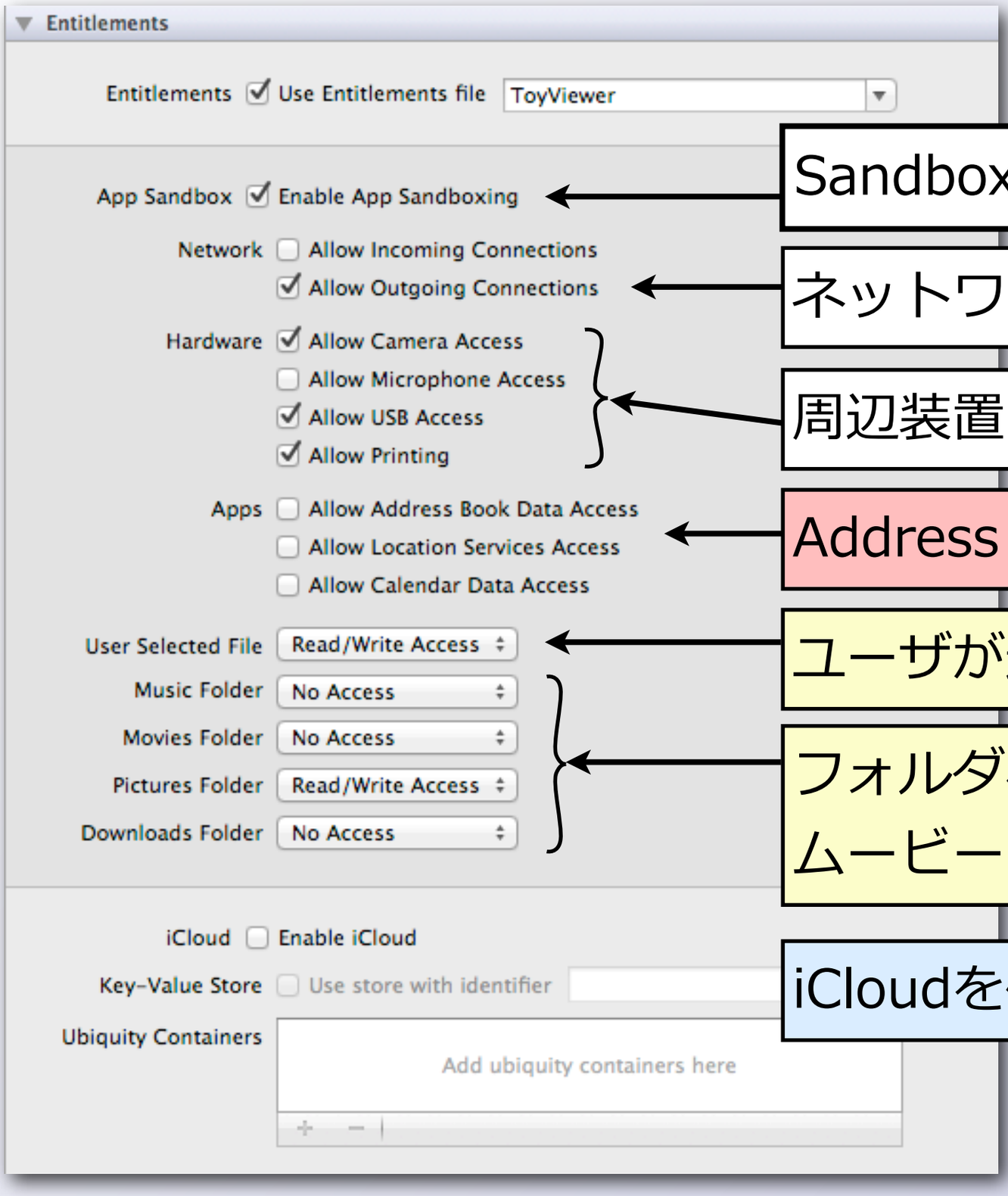
Powerbox

XPC Services

- アプリにできる機能を開発者が指定する
- プロパティリストに列挙
- Entitlementsは15個程度

署名されたアプリで有効

Entitlements



Sandboxを使う

ネットワーク：外部から／外部への接続

周辺装置：カメラ、マイク、USB、印刷

Address Book　位置情報　カレンダー

ユーザが選択したファイルへのアクセス

フォルダへのアクセス：ミュージック、ムービー、ピクチャ、ダウンロード

iCloudを使う

App Sandbox (cont.)

- アプリの動作に不可欠な、特定のディレクトリだけにアクセスを許可したい

Entitlements

Containers

Kernel Enforcement

Powerbox

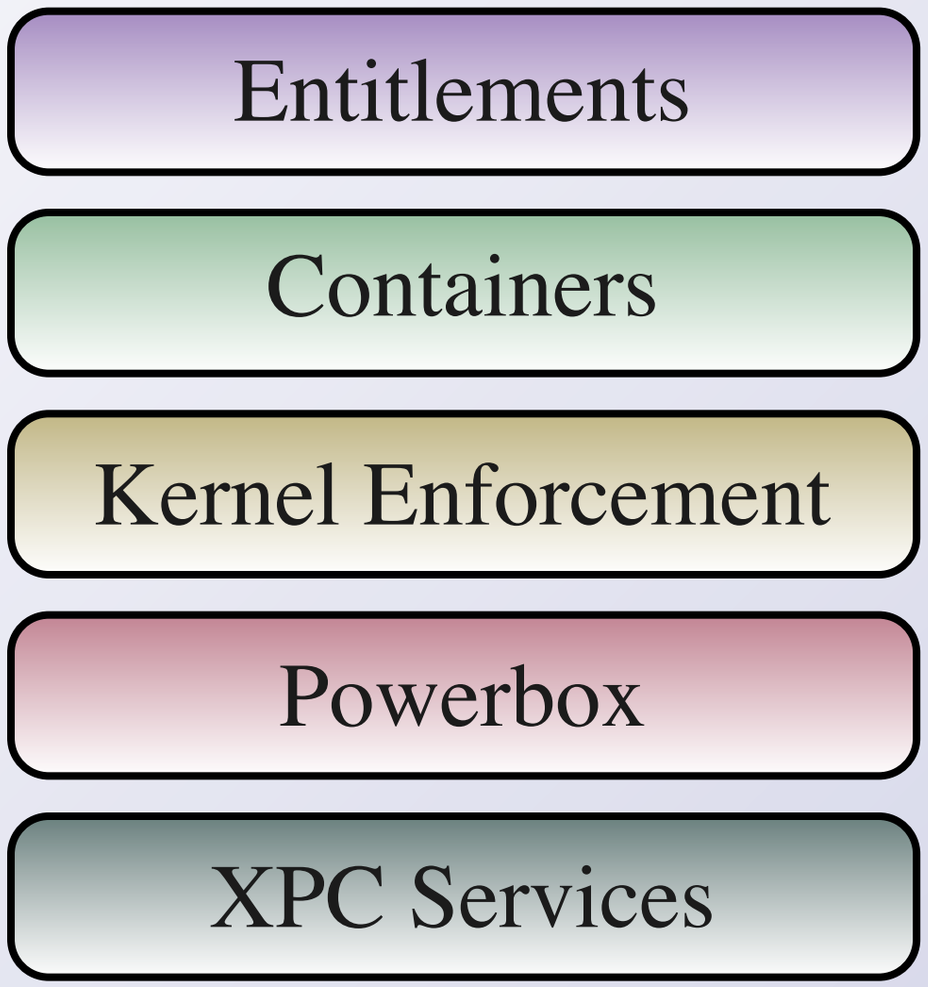
XPC Services

- ~/Library/Containers の下に、個々のアプリ専用のホームディレクトリを作る

- コンテナと特定のディレクトリ以外への直接アクセスを認めない

App Sandbox (cont.)

- アクセス制限の一方で、ユーザが指定したファイルも読み書きできないと困る



- オープンパネル／セーブパネルで指定されたファイルだけは読み書きを認める
- パス名だけからのアクセスは許さない
- 信頼できる仲介役としてPowerboxという仕組み

「最近使ったファイル」もこの仕組みを使っている

XPCとPrivilege Separation

- Privilege Separation：アプリケーションが実行できる権限(Privilege)をひとつのプロセスに集中させず、サブプロセスに任せる
  - 悪意のあるコードにアプリが制御されても、すべての権限を乗っ取られるおそれが少ない
- App Sandboxでは、サブプロセスはそのアプリ専用のデーモンとしてSandbox内で動作する
- プロセス間通信(Inter-Process Communication)の方法としてXPCが用意されている

XPCサービス

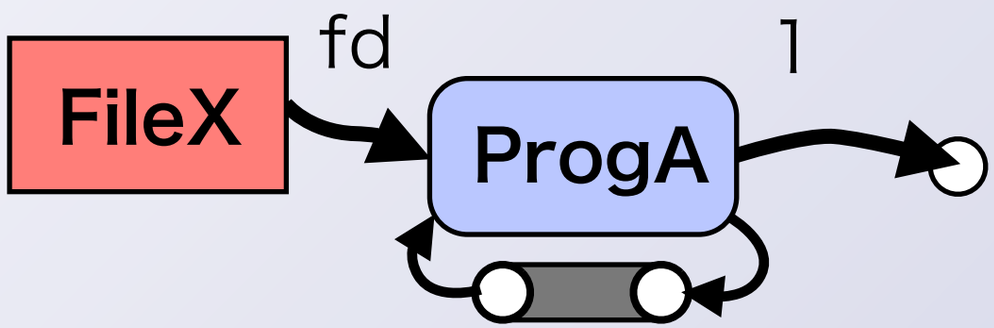
- 開発時はアプリのターゲットのひとつ
  - Xcodeでターゲットのひとつとして選択できる
- 実行時はアプリのリソースとして存在し、実行されるとアプリ専用のデーモンとして動作
  - いったん起動すると動き続け、アプリと通信しながら作業を進める
  - GCD(Grand Central Dispatch)とlaunchdで管理
- 専用のコンテナと、独自の権限を持つ
  - 単なるサブプロセスとは異なる

UNIXのパイプの概要 (1)

- 例：親プロセスが、ファイルからの入力を前処理してくれる子プロセスを起動し、その結果を処理したい

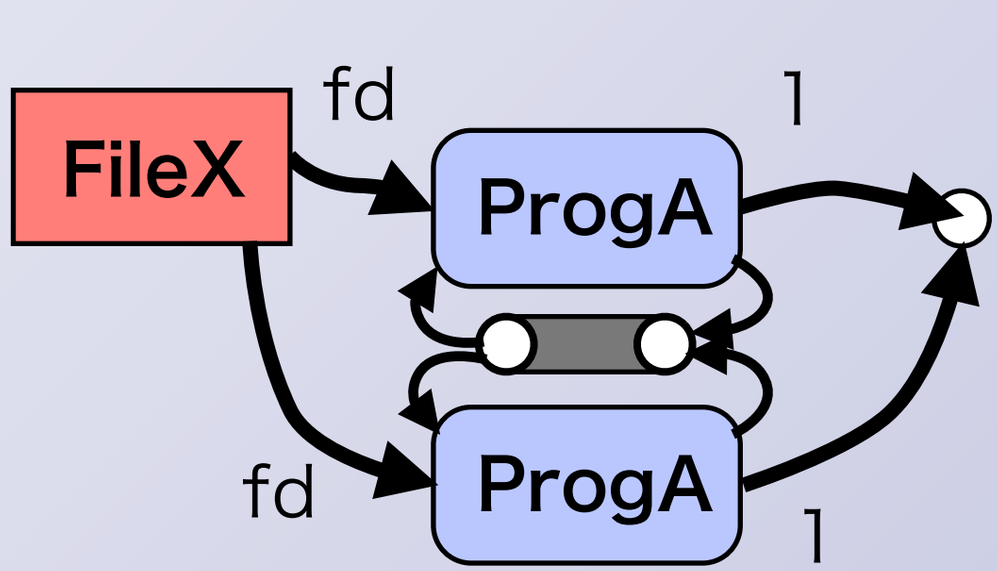
(1) まずファイルをオープン

(2) pipe() でパイプを生成する



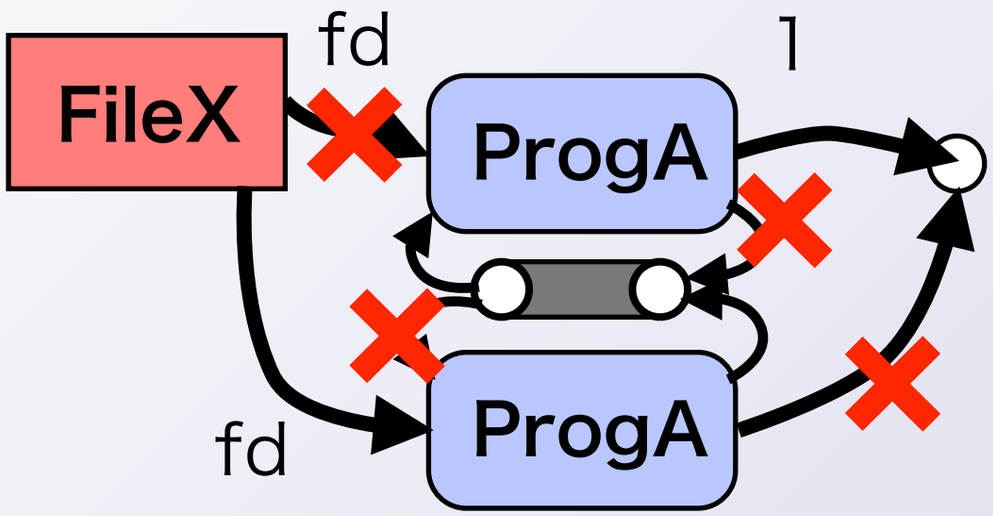
パイプは書き込み用と読み出し用のディスクリプタが対になっている

(3) fork() で子プロセスを生成

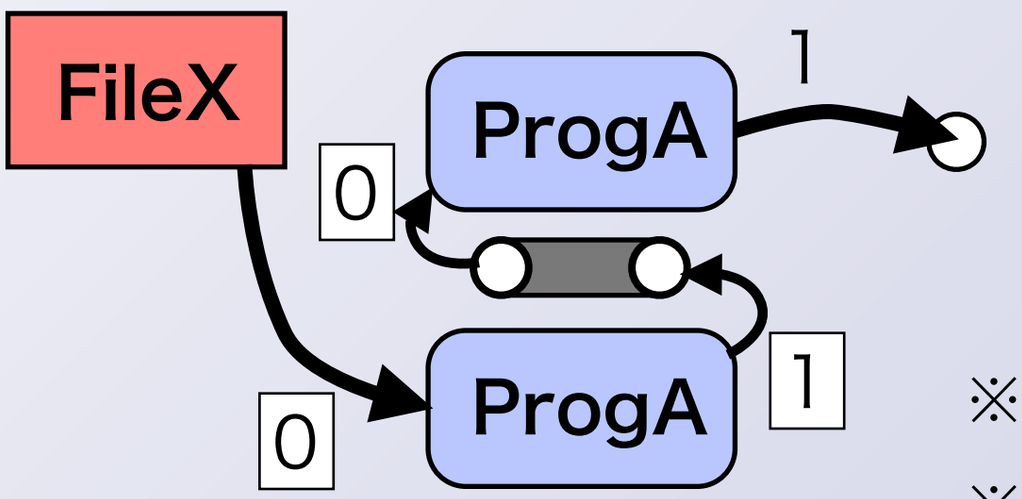


UNIXのパイプの概要 (2)

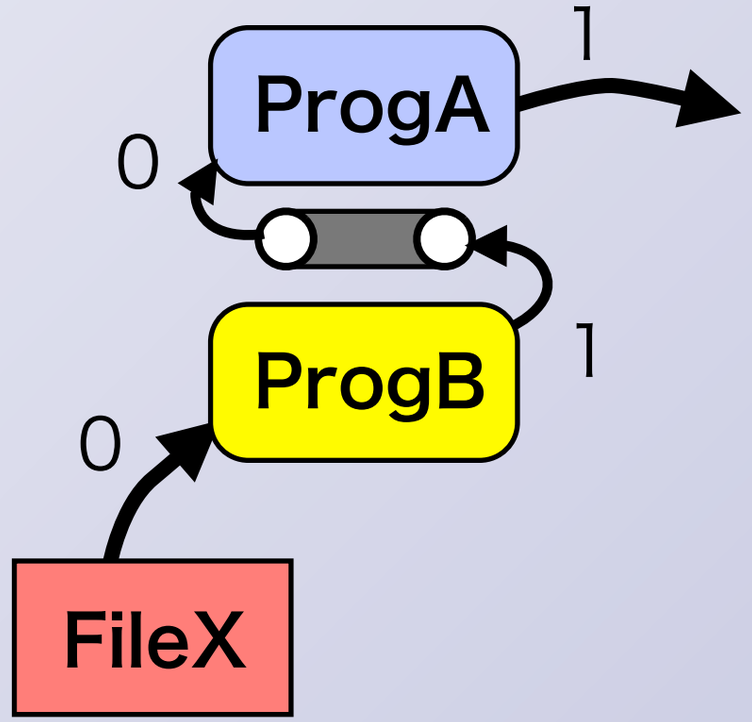
(4) 使わないディスクリプタをクローズ



(5) dup2() でディスクリプタを複製



(6) execl() で別のプログラムを
実行するプロセスになる



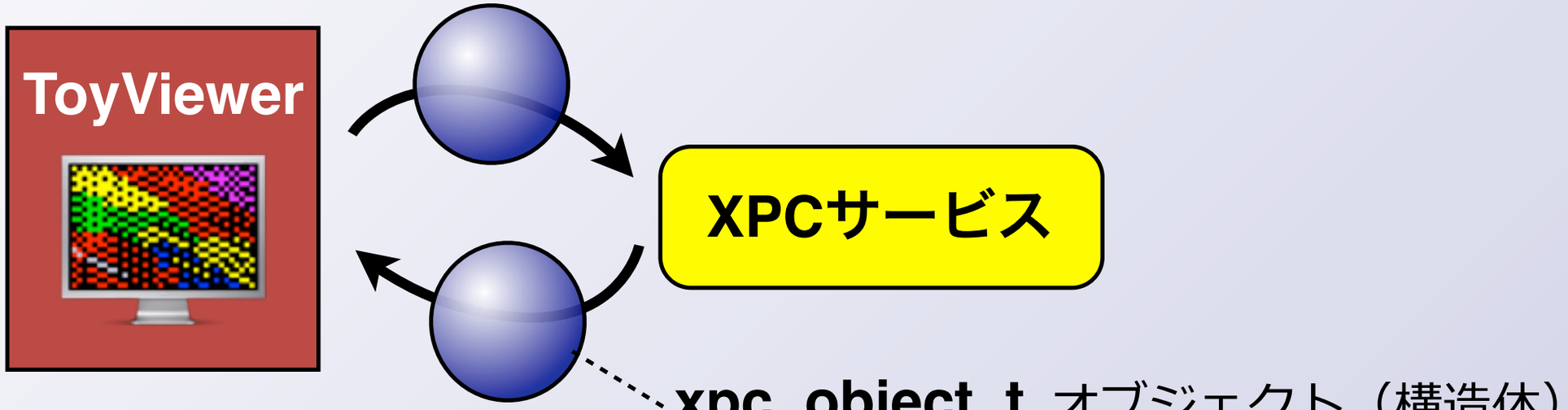
※ 似た動作は popen() 関数でも実現できる

※ CocoaではNSTaskクラスでも実現できる

XPCでフィルタを実装 (1)

- アプリ側からXPCのサービスを使う手段はあるが、パイプを使うのとは勝手が違う
  - XPCのプロセスはデーモンとしてずっと動作しているので、パイプのようにファイルディスクリプタを共有することはできない
  - Sandboxで動作していると、ファイルのパス名だけではアクセスできない
- アプリ側とXPCのプロセスの間で、メッセージをやりとりすることはできる

XPCでフィルタを実装 (2)

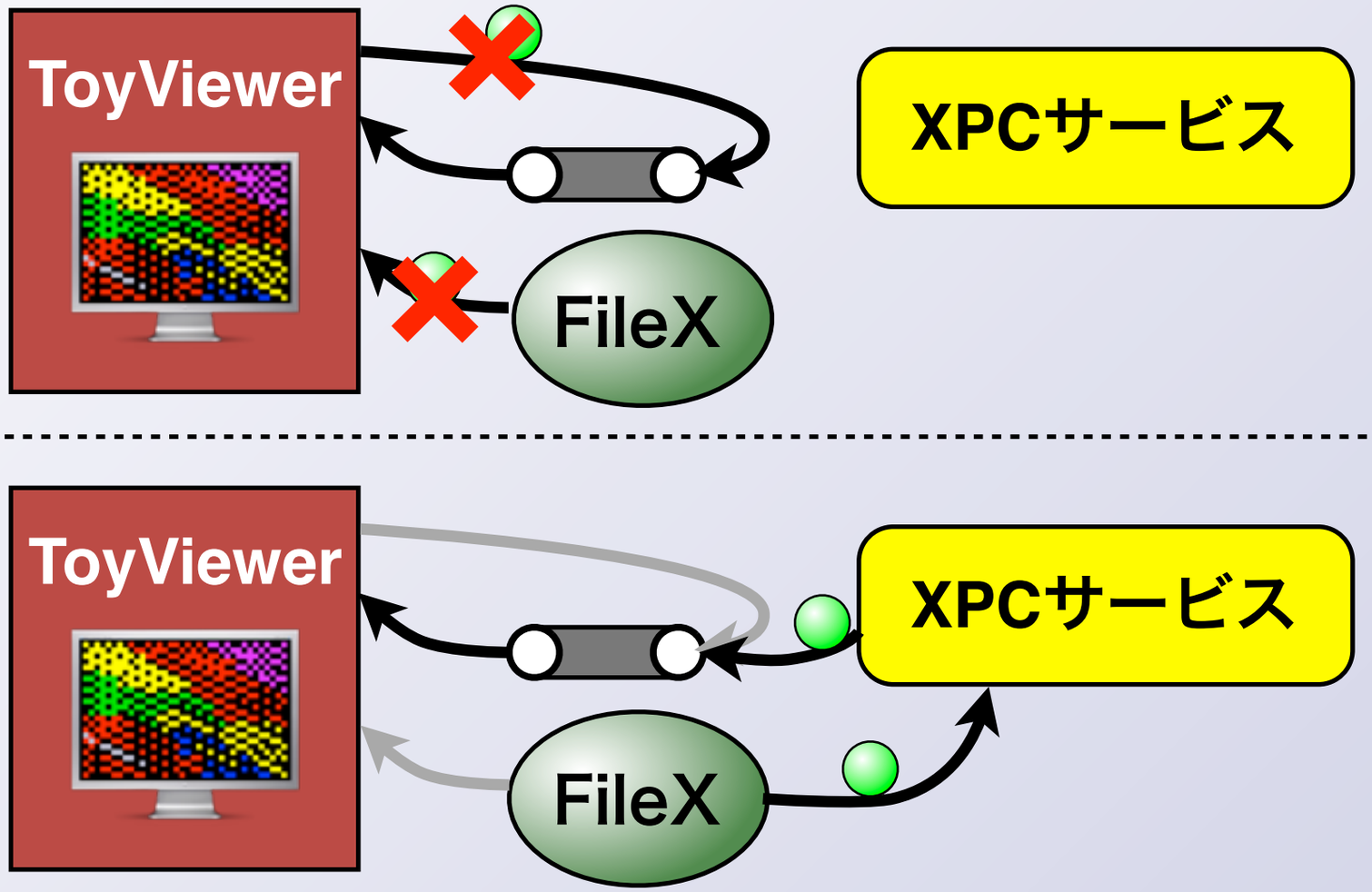


辞書オブジェクトにデータを格納して渡すことができる

整数	実数	真偽値	バイト列	文字列
オブジェクト(xpc_object_t)		日付		UUID
ファイルディスクリプタ				

- アプリ側でファイルをオープンし、そのディスクリプタを渡すと、XPCサービス側でファイルにアクセスできる

XPCでフィルタを実装 (3)



- アプリ側でファイルとパイプをオープンして、そのディスクリプタを渡すとXPCサービス側で処理できる

その他、XPCの利用に関して…

- XPCサービスのメモリ管理に注意
  - UNIXのフィルタは１つの処理ごとに終了
  - XPCサービスはずっと動作している
- メモリ管理はARCが基本
  - CベースのXPCでも、10.8からはオブジェクトをARCで管理
- サービスの依頼、結果の受け取りに注意
  - ブロック待ちではない
  - アプリ側のどのスレッドが受け取るかにも注意

まとめ

- ToyViewerの書き換えでXPCを使った
  - App Sandboxに対応するために必要
  - OSコアに近いプロセス間通信のAPIを提供
- XPCを使ってフィルタを実装できた
- XPCは進化中
  - 10.7ではC言語ベースのAPIであった
  - 10.8では NSXPCConnection などのクラスがFoundationフレームワークに追加されている
- ToyViewer自体は更新作業中・・・