I’m playing around with Screen Time on xOS again and noticed `mdls` (macOS command line utility for getting file metadata) has a `-plist` option (it probably has for a while & I just never noticed it). I further noticed there’s a `kMDItemExecutableArchitectures` key (which, too, may have been “a thing” before as well). Having application metadata handy for the utility functions I’m putting together for Rmd-based Screen Time reports would be handy, so I threw together some quick code to show how to work with it in R.

Running `mdls -plist /some/file.plist ...path-to-apps...` will generate a giant property list file with all metadata for all the apps specified. It’s a *wicked fast* command even when grabbing and outputting metadata for all apps on a system.

Each entry looks like this:

```
_kMDItemDisplayNameWithExtensions
RStudio — tycho.app
kMDItemAlternateNames

  RStudio — tycho.app

kMDItemCFBundleIdentifier
com.RStudio_—_tycho
kMDItemContentCreationDate
2021-01-31T17:56:46Z
kMDItemContentCreationDate_Ranking
2021-01-31T00:00:00Z
kMDItemContentModificationDate
2021-01-31T17:56:46Z
kMDItemContentModificationDate_Ranking
2021-01-31T00:00:00Z
kMDItemContentType
com.apple.application-bundle
kMDItemContentTypeTree

  com.apple.application-bundle
  com.apple.application
  public.executable
  com.apple.localizable-name-bundle
  com.apple.bundle
  public.directory
  public.item
  com.apple.package

kMDItemCopyright
Copyright © 2017-2020 BZG Inc. All rights reserved.
kMDItemDateAdded
2021-04-09T18:29:52Z
kMDItemDateAdded_Ranking
2021-04-09T00:00:00Z
kMDItemDisplayName
```

```
RStudio — tycho.app
kMDItemDocumentIdentifier
0
kMDItemExecutableArchitectures

  x86_64

kMDItemFSContentChangeDate
2021-01-31T17:56:46Z
kMDItemFSCreationDate
2021-01-31T17:56:46Z
kMDItemFSCreatorCode
0
kMDItemFSFinderFlags
0
kMDItemFSInvisible

kMDItemFSIsExtensionHidden

kMDItemFSLabel
0
kMDItemFSName
RStudio — tycho.app
kMDItemFSNodeCount
1
kMDItemFSOwnerGroupID
20
kMDItemFSOwnerUserID
501
kMDItemFSSize
37451395
kMDItemFSTypeCode
0
kMDItemInterestingDate_Ranking
2021-04-13T00:00:00Z
kMDItemKind
Application
kMDItemLastUsedDate
2021-04-13T12:47:12Z
kMDItemLastUsedDate_Ranking
2021-04-13T00:00:00Z
kMDItemLogicalSize
37451395
kMDItemPhysicalSize
38092800
kMDItemUseCount
20
kMDItemUsedDates

  2021-03-15T04:00:00Z
  2021-03-17T04:00:00Z
  2021-03-18T04:00:00Z
```

```
      2021-03-19T04:00:00Z
      2021-03-22T04:00:00Z
      2021-03-25T04:00:00Z
      2021-03-30T04:00:00Z
      2021-04-01T04:00:00Z
      2021-04-03T04:00:00Z
      2021-04-05T04:00:00Z
      2021-04-07T04:00:00Z
      2021-04-08T04:00:00Z
      2021-04-12T04:00:00Z
      2021-04-13T04:00:00Z

    kMDItemVersion
    4.0.1
```

We can get all the metadata for all installed apps in R via:

```
library(sys)
library(xml2)
library(tidyverse)

# get full paths to all the apps
list.files(
  c("/Applications", "/System/Library/CoreServices", "/Applications
/Utilities", "/System/Applications"),
  pattern = "\\.app$",
  full.names = TRUE
) -> apps

# generate a giant property list with all the app attributres
tf <- tempfile(fileext = ".plist")
sys::exec_internal("mdls", c("-plist", tf, apps))
```

Unfortunately, some companies — COUGH Logitech COUGH — stick illegal entities in some values, so we have to take care of those (I used `xmllint` to see which one(s) were bad):

```
# read it in and clean up CDATA error (Logitech has a bad value in one
field)
fil <- readr::read_file_raw(tf)
fil[fil == as.raw(0x03)] <- charToRaw(" ")
```

Now, we can read in the XML without errors:

```
# now parse it and get the top of each app entry
applist <- xml2::read_xml(fil)
(applist <- xml_find_all(applist, "//array/dict"))
## {xml_nodeset (196)}
##  [1] \n  _kMDItemDisplayNameWithExtensions\n  1Blocker (Old).app\n
kMDItemAlternateNames\n ...
##  [2] \n  _kMDItemDisplayNameWithExtensions\n  1Password 7.app\n
_kMDItemEngagementData\n   ...
##  [3] \n  _kMDItemDisplayNameWithExtensions\n  Adblock Plus.app\n
```

```
kMDItemAlternateNames\n   ...
##  [4] \n  _kMDItemDisplayNameWithExtensions\n  AdBlock.app\n
kMDItemAlternateNames\n  \n  _kMDItemDisplayNameWithExtensions\n
AdGuard for Safari.app\n  kMDItemAlternateNames\n  _
kMDItemDisplayNameWithExtensions\n  Agenda.app\n
kMDItemAlternateNames\n  \n  _kMDItemDisplayNameWithExtensions\n
Alfred 4.app\n  kMDItemAlternateNames\n  \n  _
kMDItemDisplayNameWithExtensions\n  Android File Transfer.app\n
kMDItemAlternateNames< ...
##  [9] \n  _kMDItemDisplayNameWithExtensions\n  Asset Catalog Creator
Pro.app\n  kMDItemAlternateNa ...
## [10] \n  _kMDItemDisplayNameWithExtensions\n  Awsaml.app\n
kMDItemAlternateNames\n  \n  _kMDItemDisplayNameWithExtensions\n
Boop.app\n  kMDItemAlternateNames\n  \ ...
## [12] \n  _kMDItemDisplayNameWithExtensions\n  Buffer.app\n
kMDItemAlternateNames\n  \n  _kMDItemDisplayNameWithExtensions\n  Burp
Suite Community Edition.app\n  kMDItemAlternat ...
## [14] \n  _kMDItemDisplayNameWithExtensions\n  Camera Settings.app\n
kMDItemAlternateNames\ ...
## [15] \n  _kMDItemDisplayNameWithExtensions\n  Cisco Webex
Meetings.app\n  kMDItemAlternateNames\n  _
kMDItemDisplayNameWithExtensions\n  Claquette.app\n
kMDItemAlternateNames\n  \n  _kMDItemDisplayNameWithExtensions\n
Discord.app\n  kMDItemAlternateNames\n  \n  _
kMDItemDisplayNameWithExtensions\n  Elgato Control Center.app\n
kMDItemAlternateNames< ...
## [19] \n  _kMDItemDisplayNameWithExtensions\n  F5 Weather.app\n
kMDItemAlternateNames\n  \n _kMDItemDisplayNameWithExtensions\n
Fantastical.app\n  kMDItemAlternateNames\n  < ...
## ...
```

I really dislike property lists as I’m not a fan of position-dependent records in XML files. To get values for keys, we have to find the key, then go to the next sibling, figure out its type, and handle it accordingly. This is a verbose enough process to warrant creating a small helper function:

```
# helper function to retrieve the values for a given key
kval <- function(doc, key) {

  val <- xml_find_first(doc, sprintf(".//key[contains(.,
'%s')]/following-sibling::*", key))

  switch(
    unique(na.omit(xml_name(val))),
    "array" = as_list(val) |> map(unlist, use.names = FALSE) |>
map(unique),
    "integer" = xml_integer(val),
    "true" = TRUE,
    "false" = FALSE,
    "string" = xml_text(val, trim = TRUE)
  )
```

```
}
```

This is nowhere near as robust as `XML::readKeyValueDB()` but it doesn't have to be for this particular use case.

We can build up a data frame with certain fields (I wanted to know how many apps still aren't Universal):

```
tibble(
  category = kval(applist, "kMDItemAppStoreCategory"),
  bundle_id = kval(applist, "kMDItemCFBundleIdentifier"),
  display_name = kval(applist, "kMDItemDisplayName"),
  arch = kval(applist, "kMDItemExecutableArchitectures"),
) |>
  print() -> app_info
## # A tibble: 196 x 4
##    category        bundle_id                             display_name
arch
##
##  1 Productivity    com.khanov.BlockerMac                 1Blocker
(Old).app
##  2 Productivity    com.agilebits.onepassword7            1Password
7.app
##  3 Productivity    org.adblockplus.adblockplussafarimac Adblock
Plus.app
##  4 Productivity    com.betafish.adblock-mac              AdBlock.app
##  5 Utilities       com.adguard.safari.AdGuard            AdGuard for
Safari.app
##  6 Productivity    com.momenta.agenda.macos              Agenda.app
##  7 Productivity    com.runningwithcrayons.Alfred         Alfred 4.app
##  8 NA              com.google.android.mtpviewer          Android File
Transfer.app
##  9 Developer Tools com.bridgetech.asset-catalog          Asset
Catalog Creator Pro.app
## 10 Developer Tools com.rapid7.awsaml                     Awsaml.app
## # … with 186 more rows
```

Finally, we can expand the `arch` column and see how many apps support Apple Silicon:

```
app_info |>
  unnest(arch) |>
  spread(arch, arch) |>
  mutate_at(
    vars(arm64, x86_64),
    ~!is.na(.x)
  ) |>
  count(arm64)
## # A tibble: 2 x 2
##   arm64     n
##
## 1 FALSE    33
## 2 TRUE    163
```

Alas, there are still some stragglers stuck in Rosetta 2.