# Attachment Converter Workshop, iPres 2023

Nishchay Karle, Obi Obetta, Matt Teichman

September 18, 2023

Welcome to the Attachment Converter Workshop at iPres 2023! In this session, we'll walk you through our new open source application, Attachment Converter, which batch-converts all attachments in an email mailbox to preservation formats.

Then next few sections of the handout include background on the project for your reference, but when the workshop starts, we'll be working through the material on this handout starting with section 4.

## 1 Project Website

The project website is located here:

https://dldc.lib.uchicago.edu/open/attachment-converter

## 2 How to Participate in This Workshop

There are two ways to participate in this workshop. If you're feeling tech-savvy, we would encourage you to install the required software in advance of the workshop and type along with us as we walk through some illustrative examples of the email format. If you aren't feeling tech-savvy, you should be able to just watch and follow along. Either way, we are really looking forward to engaging with your questions and comments as we show you how to use our new tool. If you aren't sure how tech-savvy you're feeling, the question to ask is whether you're comfortable opening the Terminal application on your computer and working at the command prompt.

In the next section, we'll go through how to install the software you'll need if you want to participate in the workshop by typing along on your own machines. If you're planning to simply attend, watch, listen, and ask questions, please feel free to skip to section 4, which is what we'll be working off of during the workshop—you won't need to set anything up on your computer in advance.

# 3 Advance Preparation

If you're planning to type along with us on your computer during the workshop, then this is the section for you!

The software you'll need to install for the workshop is slightly different, depending on whether you're working in Windows or macOS. Either way, you will need to have privileges on your machine that allow you to install software, so if you're attending this conference from a work machine, that might be something worth looking into with your system administrator.

If you're on Windows, please skip to section 3.2. If you're on a Mac, you can proceed to section 3.1.

## 3.1 macOS

If you're on a Mac, you'll need to open a Terminal, then install an open-source package manager, the `git` version control system, the GNU Make build tool, and the `libpst` package.

Remember: to follow these instructions, you'll need to have the ability to install software on your machine, so if you don't, you may want to reach out to your system administrator to see whether they can grant you the appropriate privileges for doing so.

### 3.1.1 Install Homebrew

There are various options for open source package managers on macOS, but we recommend using Homebrew. If you've never used it before, you'll first need to install XCode Command Line Tools, which you can do by running this command in your Terminal:

```
$ xcode-select --install
```

Then you can install Homebrew by following the instructions here:

https://brew.sh/

Or, equivalently, by typing this command:

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

### 3.1.2 Install Libpst

The last thing we'll ask you to install is Libpst, the software we will use to convert from Outlook `.pst` to MBOX format during the workshop. To install that, run:

```
$ brew install libpst
```

Once you've reached this point on your Mac, you can skip the next section—which is our Windows-specific setup instructions—and proceed straight to section 3.3.

## 3.2 Windows (Debian WSL)

Attachment Converter is a UNIX application, which means that in order to run it on Windows, you'll need to install the Windows Subsystem for Linux. We chose Debian as a Linux distribution for this purpose, because Debian has full out-of-the-box support for OCaml, the programming language that Attachment Converter was written in.

So first, you'll install the Debian WSL. Once that's set up, you'll open up a Debian WSL Terminal and do everything else from inside that Terminal, including installing a few more utilities, as well as running Attachment Converter itself.

Note that you need to have privileges to install software on your machine to follow these instructions. If you don't, check with your system administrator about how to get them.

### 3.2.1 Set up the Debian WSL

To set up the Debian WSL:

- open up the Microsoft Store application using your Start Menu
- there should be a search box at the top of the window that opens
- type "Debian" in the search box and hit Enter
- a list of search results will come up; when you find the one called Debian with an icon that looks like this, click on "Get"
- after it's finished installing, to open a Debian WSL terminal, run "Debian" form the Start Menu
- when the terminal first opens up, you will be asked to choose a username and password for your Linux subsystem
- when you're typing your password, it won't show anything, but you will still be typing it
- don't forget to write those credentials down and keep them available for reference
- that will probably be the end of the install process, but if it asks you to reboot, do that

Once you've completed the above steps, if your Debian WSL terminal is not already open, you can open it by choosing "Debian" from the Windows Start Menu. If it asks you to log in, use the username and password you chose during the installation process.

### 3.2.2 Prep WSL for installing

Before installing everything to your WSL, it will be necessary to synchronize your machine's installation with the website you're going to download software from. To do that, first run this command:

```
$ sudo apt update
```

You should see a bunch of information get printed to the screen about it connecting to some websites and downloading some information. It should also ask you to type the password you chose during the Debian WSL installation process, since this is the first time you're running an install command.

Next, the WSL needs the latest version of all the software it came pre-installed with. To install all of those software packages in one go, run this command:

```
$ sudo apt upgrade
```

(Similar to the previous command, but it says `upgrade` instead of `update`.) As always, you'll see a bunch of information get printed to the screen. If it prompts to say yes, say yes.

### 3.2.3 Create Installation Directory

Next, you need to create the directory the Attachment Converter program is going to get installed to, which you can do by running these commands:

```
$ cd ~
$ mkdir bin
```

### 3.2.4 Install Version Control Software

Now that your UNIX environment is set up, the next step is to install version control software, which in this case is Git. To do that, run this command:

```
$ sudo apt install git
```

This is the utility we will use to get the latest version of the source code for Attachment Converter, later in these setup instructions.

If it asks you for your password, use the one that you chose when installing Debian WSL. If it asks you to confirm you want to install Git, say yes. (You'll be saying yes to everything that comes up in these instructions)

#### 3.2.5 Install GNU Make

The software we're going to use to compile Attachment Converter is called Make. To install it, run this command:

```
$ sudo apt install make
```

#### 3.2.6 Install Libpst

Finally, we're going to ask you to install Libpst, which is a freely available utility for converting Outlook `.pst` files to MBOX format—the email mailbox format that Attachment Converter uses. To install it:

```
$ sudo apt install libpst4
```

Once you've reached this point on your Windows machine, you're ready to go to the next section, in which we show you how to compile Attachment Converter into an executable that you can run.

### 3.3 Compile Attachment Converter

Now that you're set up with the basic software you need, whether you're on Windows or a Mac, the next step is to download the source code for Attachment Converter, compile it into an executable you can run, and put the executable in a location where your Terminal can see it.

#### 3.3.1 Get The Code

The first thing we need to do is download the source code for Attachment Converter. The simplest way to do that is by using Git.

First, make a new directory to keep your source code in by running these commands:

```
$ cd ~
$ mkdir src
$ cd src
```

To then download the source code for Attachment Converter using Git, run this command (you can copy/paste it if it's too long to type):

```
$ git clone https://github.com/uchicago-library/attachment-converter.git
```

That will download all the source code and put it into a directory called `attachment-converter` under `src`. To go into that directory, type:

```
$ cd attachment-converter
```

As an aside, if you're on Windows and want to view the contents of a directory you're in using Windows explorer, you can run this command to open up an Explorer window in the current directory (note the dot after the `explorer.exe` command):

```
$ explorer.exe .
```

If you're on a Mac, you can do the same thing—i.e. view the directory you're in in Finder using the `open` command:

```
$ open .
```

Now that you have the source code for Attachment Converter, the next step is to compile it into an executable program.

### 3.3.2 Compiling, the Semi-Automated Way

Let's start with an overview. The utility we're going to use to compile Attachment Converter is called Make. If you're on Windows, we told you to install that in the previous section. If you're on a Mac, then you already have Make installed on your computer.

Attachment Converter has a lot of moving parts, which means that installing it involves installing some more standard utilities and copying a lot of different files to a lot of different places in your home directory. When you run Make, the full list of things it will do is:

- install all the free software that Attachment Converter uses to convert file attachments

- install `opam`, the package manager for the OCaml programming language
- create a location in your home directory for all `opam` files to go in
- install `dune`, the OCaml build tool, to that location
- install all third-party OCaml libraries that are necessary to compile Attachment Converter
- put a number of different configuration files in places where Attachment Converter expects them to be, in order to run

To compile Attachment Converter and then install it, first make sure you're in the `attachment-converter` directory, which is where Git downloaded and put all of the code:

```
$ cd ~/src/attachment-converter
```

Then from the `attachment-converter` directory, run:

```
$ make home-install
```

You'll see many messages get printed to the screen, and it should generally look like it's downloading and installing various programs, displaying progress bars, and so forth. This is your cue to go heat up a pot of tea, because it should take about 5-10 minutes. The process may pause at one point to ask you to type in your administrator password, in which case you should use the one you chose when you installed Debian. You may also be prompted to confirm certain steps with a yes/no prompt; if that happens, just choose "yes" each time. There will also be one or two times when it won't display anything on the screen, even though it's still working. You'll know it's done when you see the final confirmation message.

When the installation process is done, it should print a message that looks like this:

```
Attachment Converter has been installed to ~/bin/attc.
Please ensure that ~/bin is on your path.
```

Once the installation process is finished, `~/bin` needs to be on your shell path in order for Attachment Converter to run. If you don't know what that means, run this command if you're on Windows:

```
$ echo "export PATH=~/bin:$PATH" >> ~/.bashrc
```

And run this command if you're on a Mac:

```
$ echo "export PATH=~/bin:$PATH" >> ~/.zshrc
```

Then close and reopen your Terminal.

#### 3.3.3 Compiling, the Manual Step-By-Step Way

We've put a lot of work into making the semi-automated installation process via Make work, but it's complicated and there is always some chance it will throw an error. If you get an error while running Make, another thing you can try is to do all the steps that our Make configuration does indivdually. Following all these steps should work, if there's an unexpected error in our Make configuration. (Though if you do encounter an error, we would love to hear about it, so that we can fix it and update these instructions!) Installing Attachment Converter in that way will probably take you a bit longer.

The full instructions for setting Attachment Converter up in the non-automated way can be found on our website here:

https://dldc.lib.uchicago.edu/open/attachment-converter/docs/

That concludes our setup instructions! The rest of this handout reflects what we will cover during the workshop proper.

## 4 During The Workshop

Welcome to our workshop! We are excited to be here.

Attachment Converter is a command-line utility that batch-converts all attachments in an email mailbox to preservation formats. You give it your email in the form of an MBOX file, and it creates a new MBOX file with copies of all the attachments in preservation formats, next to the original attachments in the emails from which they originated.

Let's open the workshop with a quick demo of Attachment Converter.

### 4.1 Quick Demo

In this demo, we:

- run Attachment Converter on a small example MBOX containing five emails
- the example MBOX contains attachments in the following formats:
  - DOC
  - DOCX
  - JPEG
  - PDF
- those attachments are then converted to, respectively:

- TXT, PDF-A-1b
- TXT, PDF-A-1b
- TIFF
- PDF-A-1b

# 5 Background

You're most likely used to using email clients, whether they're web-based, like GMail or Hotmail, or run as apps on your computer, like Thunderbird, Apple Mail, or Outlook. But what does an email actually look like, close up?

Interestingly, the email format is not only very old, totally ubiquitous, and mostly standardized, but it is actually human-readable! At the level at which mail servers send and receive mail, every email is in fact plaintext—that is to say, standard ASCII characters with no fonts, styling, sizing, or page layout information in them of the kind you see in word processors. With most other software, if we wanted to look at the data it was sending around, it would be tricky, because it would be raw binary data. But with email, the raw data are just sequences of characters you could read yourself, if you wanted to.

The format of an individual email is pretty standardized, but there are many different data formats for putting a large collection of individual emails together into a *mailbox*, such as **Inbox**, **Sent Mail**, or **Trash**. Attachment Converter uses one of the oldest and most universal data formats for mailboxes, called MBOX.

## 5.1 The MBOX format

Attachment Converter works with mailboxes in MBOX format, which is the form in which GMail gives you your email when you request a local backup from your GMail account.

MBOX is an old, standard, and human-readable format. In other words, rather than packing large collections of individual emails into a raw binary data format, the mailbox containing emails is itself also plaintext. So in the same way that you can open the full data in an email up in any text editor, you can open an MBOX up in a text editor and just look at the information that's in there.

The MBOX format is very simple. One thing that makes it simple compared to other formats is that it saves each mailbox in a single file. That makes it easy to browse through large sets of mailboxes, move them around, back them up, and so forth. Another thing that makes it simple is that it's nothing other than a format for putting emails into a sequence. So an MBOX is essentially a big list: one email followed by another until you're through all of them. This is as opposed to e.g. trying to group/organize the emails in some way, or trying to include

information about what emails are contained in it. (Later on in this workshop, we'll demonstrate Attachment Converter's "report" feature, which you can run on an MBOX when you're browsing around to get some basic information about it.)

### 5.2 The Delimiter: `From`

Any data arranged into the form of a list on a computer needs some way of specifying where each item in the list starts and where it finishes. Usually, the way we accomplish that is by using a *delimiter*. For example, if I were to write down the list "`1,2,3`" as a string of characters, the delimiter in that example would be a comma and the elements of the list would be `1`, `2`, and `3`, respectively.

In the case of an MBOX, each email begins with a special line of text that is not considered to be part of the email—only part of the mailbox. The rules for constructing a From line go like this:

- put `From` at the beginning of the line
- add one space
- insert any text you want (i.e. this part is a free text field)
- end the line

So the following are all perfectly good From lines:

- `From Matt Teichman`
- `From MAILER-DAEMON Fri Jul 8 12:08:34 2011`
- `From ...malomadingdong`
- `From vd7g8o73 2vfy^&32v///\\7y329?~‘“‘xxx`
- `From` (with a space before the line break)

One thing that's potentially confusing about From lines is that emails typically come with a header telling you who the sender was. That header usually looks like this:

```
From: Bugs Bunny <bugsbunny@uchicago.edu>
```

A header like that is part of an individual email—not the delimiter in a mailbox—and a quick way to tell which of these two things you're looking at is to look for a colon. So if you see `From` with just a space, it's the MBOX delimiter, and if you see `From:` with a colon, it's an email header.

So a mailbox in MBOX format is just a sequence of emails with a From line before each one. Notionally, it looks like this:

```
From MAILER-DAEMON Fri Jul 8 12:08:34 2011
(first email goes here)
From MAILER-DAEMON Fri Jul 8 12:08:34 2011
(second email goes here)
From MAILER-DAEMON Fri Jul 8 12:08:34 2011
(third email goes here)
```

And so on. We conclude this section by opening up our example mailbox.

### 5.3 The Anatomy of an Email

The email specification is very, very complex and has also evolved a great deal since the technology first emerged in the early 1970s. It would take longer than one workshop session to cover all the details, so what we will instead do is focus on the parts of the email specification that are most relevant to what Attachment Converter does.

Fundamentally, an email consists of *headers* followed by a *body*. The headers are separated from the body by two line breaks. The headers function like metadata for an email; they provide an informational summary about what's in the email either to the recipient or to the recipient's email software. The body is the main part of the email, as in the part the recipient is meant to see.

It might surprise you to hear that you can have an email with no body, but if you think about it, every time you accidentally hit send before typing anything could be a case of that, depending on how your email software decides to do it. But although you can have emails with no body, you can't have an email with no headers. The two required headers are a date header and a from header, they don't have to come in any particular order, and they look like this:

```
Date: Tue, 10 Aug 2004 14:17:45 -0500
From: Daffy Duck <daffyduck@uchicago.edu>
```

It's also possible to have a body with just text and no attachment. In that case, the body of an email looks like what you'd expect:

```
Dear Road Runner,

For my whole life, I've wondered what the Warner Brothers foley
artists might have done to create that sound of you blowing a
raspberry with your tongue.  My best guess is that it's the sound of a
```

```
person moving their hand over the top of a wet Coke bottle, but I
can't be sure.  If there is anything you could do to clear this
mystery up for me, I would be eternally grateful.

Yours truly,
Porky Pig
```

In that bare bones type of email, that's it! When you're finished the text, you've reached the end of the email. No attachments yet. If we want to start having attachments, we need to look at an extension to the original email specification that was created in the early 1990s, called Multipurpose Internet Mail Extensions, or MIME.

### 5.3.1 MIME types

Normally when we talk about attachments, we think of e.g. a file that you selected from your computer that you are sending to someone along with an email. But an attachment can also just be another email—for example, that's what a lot of email software does when you forward an email to someone else: you write what you're going to write, but then attached to your comment on the forwarded email is the email you are forwarding. Believe it or not, in an even more exotic type of case, you can put a file straight into the body of an email, not as an attachment.

This is relevant for our purposes, because insofar as the email specification is concerned, attached emails and attached files enjoy equal status as attachments, even though colloquially we tend to assume that an attachment is a file you're loading into an email from your computer. We will deal with this potential terminological confusion by continuing to use the term *attachment* to specifically mean *file attachment* in contexts where it is clear, but insisting on the term *file attachment* where there is ambiguity.

Like the email specification, the MIME specification is incredibly complicated, and allows you to do many, many different things with emails. In this workshop, we are going to focus on the fact that MIME allows you to write an email with multiple parts, each of which are bodies of an email. So, for example, when you write an email to your friend with a note in the body, followed by a file you want to send to them, that is the format in which it will get sent out.

Let's focus on that pretty ordinary type of case. After the two required headers, the from header and the date header, there will be the following MIME version header, which is required if you're using MIME:

```
MIME-Version: 1.0
```

After that, you'll usually find the following optional but bog-standard content type header, which tells you what kind of MIME-encoded data you're looking at:

```
Content-Type: multipart/mixed;
          boundary=name-of-boundary-someone-chose
```

That tells you that what you will be looking at next is a sequence of MIME-encoded email bodies, and what the delimiter for that sequence of email bodies will be. If there are no more headers in the main email, then the list of headers will end with two line breaks, followed by a delimiter:

```
--name-of-boundary-someone-chose
```

Note that the delimiter is whatever the application creating the email chose to name it, under the `boundary` header field parameter in the content type header, preceded by two hyphens `--`. The final delimiter in the sequence will have two additional hyphens at the end of it, before the line break.

Back to the first delimiter. Following the first delimiter will be the list of MIME headers that provide information about the first email body in the multipart list. What headers are those? There could be a lot of them, but there will normally at least be a content type header:

```
Content-Type: text/plain
```

After the MIME headers for the first email body in the list are finished, there will be two line breaks, followed by the message you wrote to your friend, followed by another boundary:

```
Dear Foghorn Leghorn,

Have you ever wondered there is more to life than being a cartoon?
I've wondered about this ever since I first realized I wasn't real
during the short animation, Rabbit Rampage.  Eager to hear your
thoughts.  I'm attaching a book on existential phenomenology by Simone
de Beauvoir that I think might help.

Yours truly,
-Elmer Fudd

--name-of-boundary-someone-chose
```

The boundary tells us that we're moving onto the second part of the MIME multipart body, which is a second email body. This will be the file attachment, which we look at in the next section.

### 5.3.2 Base64 data

You might be surprised to hear that when you attach a file to an email, it isn't a file anymore. That is, it isn't some data sitting on your hard drive, physically arranged on disk into whatever types of blocks your operating system understands, the way files on your computer all are. The data within a file are nothing more than a sequence of numbers, and when you move that sequence of numbers off your hard disk and into an email, the sequence goes straight into the email. However, that sequence of numbers doesn't go into the email in its raw form; if it did, then your text editor would likely get confused when you tried to open it, and everything would probably look bizarre. So under MIME, what we do is convert the data in your file into a plaintext representation, in which every number in the sequence is converted to a printable character that you can view in a text editor.

There are a few ways of doing that conversion, but the one we're going to focus on is Base64 encoding. Base64 encoding turns raw binary data into something that's pretty compact, and also pretty painless to look at. Here's what it looks like:

```
JVBERi0xLjcKJeLjz9MKMTU1IDAgb2JqCjw8L0ZpbHRlci9GbGF0ZURlY29kZS9MZW5ndGggMTI1
Nz4+c3RyZWFtCnjavVjbjts2EH3XV+gLZN4vwGKBbDYpAjQPbfwDthwDabtpG+T/UQO5JIeySCtp
EOwKulEzZ25nhj68+vL10/U0fx2f3r8e/h3YCH+//zJIM1rnJz2+DFp7vP5r+DD8NrxZVh4+/HP6
PD48HN6/fvc8ssfH8ekZvn86Doe3fORiPF6DDCMm5sbjZXhgTGrGlGNM++U84/Xp8fjH8Oa4KZdv
y7V2YqoWzWRHjGjDU3fhxbOyoGP5X+61WA5Y+3F5dl6uzXJe1vPLHRxyGwdXdvIrKIs6dY6qQa0A
KOkZQDOoVkY4yuM6tdwvl0rjOoDr4jV8rwEmj2thDdwHswyuYWgeL7rB9IwDXaRUlAuuCvrn6JKw
XuH3tv4G3ofnCt1nwVX3YqfasZOsOEx4FOqic8BRwWCWAHV06G0dIe+pki5Q0wbK2f0koxG5m0i2
kUhMTV6t9Bl0AMNMgOOKkbmWbBDnEhltO7pdQ7fnE+e17q4Nvukv41f4Q0ZiBoXguh0+4g1S4sBw
k+BCm5GPjk9aZ40vQaMkdXZF7QkFvkt1s871/K2Lz8M3rL5PMqE2Uy2DnBSVnCGG1Fqy/NyzuEGX
```

. . . and so on, for pages and pages, depending on how big the file you Base64-encoded is. Without going too far into the details of the encoding, the way it works is that each character in the above representation stands for a number between 0 and 63. So you can represent the data inside of any file as a stream of readable characters.

This gets us to the next part of our hypothetical email. After the end of the first body in the MIME multipart (in this example, the note from Elmer Fudd), we get some MIME headers that describe the second part:

```
Content-Type: application/pdf
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename*=utf-8''beauvoir.pdf;
        filename="beauvoir.pdf"
```

The content type header tells us that what we're about to look at is a PDF. That plus the fact that the content disposition says `attachment` tells us that it is a file attachment. The content disposition header also contains additional header field parameters called `filename` and `filename*`, respectively, which is what your email software is going to use to determine the name of the file when the recipient saves it to their machine. Annoyingly, the `filename*` header field parameter is not part of the MIME specification—the MIME specification just says that your email software can create any header field parameters it chooses, and also that it can have as many of them as your email software chooses to create. But although it is not actually part of the MIME specification, most email software you'll use will nonetheless look for that header field parameter and use it to determine the attachment's file name for when it gets saved to the recipient's machine.

After the MIME headers for the attached PDF will be the Base64-encoded data of the PDF itself, which will look like the above text and could take up many pages, depending on how large the original PDF was. Although many mail user agents such as GMail impose a 25-megabyte restriction on attachments, the email specification itself imposes no such restriction.

After the data for the PDF ends, there will be two line breaks, followed by one final MIME boundary:

```
--name-of-boundary-someone-chose
```

That concludes our hypothetical example of an email with a prose body and one file attachment, whose body is a multipart MIME with one `text/plain` email body and one PDF file attachment body.

### 5.3.3 How To Convert an Outlook `.pst` to `.mbox` Format

At the University of Chicago, we are usually given mailboxes in one of two forms for accessioning: MBOX and Outlook PST. In order to convert the attachments that occur in our PST collections, we use an application called `readpst` to convert them.

Assuming your input `.pst` file is called `mailbox.pst` and that you are currently in the directory where `mailbox.pst` is located, then the command to run `readpst` is as follows:

```
$ readpst mailbox.pst
```

After you run that command, `readpst` will put a copy of the original `.pst` file into the same directory you ran it from, called `Inbox.mbox`. To rename the file to something more reasonable, run this command:

```
$ mv Inbox.mbox mailbox.mbox
```

Now the original `.pst` and the converted `.mbox` have the same basename.

## 5.4 More Detailed Demo

In the next part of the workshop, we'll redo the demo we did at the beginning, knowing what we now know about how emails are structured and how MIME multiparts allow us to attach files to them.

We begin by installing Attachment Converter. For more info on that, please see our pre-workshop installation instructions. After the installation, we'll run Attachment Converter once again on the example MBOX we used during the initial demo.

Next, we'll take a look at what Attachment Converter did to the attachments in our example MBOX.

### 5.4.1 Looking At The Output

We saw during the demo earlier that when we opened the converted MBOX in Apple Mail, the attachments appeared next to the originals. What we are now in a position to do is open the converted MBOX in a text editor and see what it really did, looking at the raw data.

Let's start with the first email, featuring the `.doc` attachment. We know it's going to be a MIME multipart because it features these headers:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="=-=-="
```

We can see from this content type header that the boundary for the multipart will be `=-=-=`. If we scroll a bit further, we can see the first part of the MIME multipart is the prose part of the email:

```
--=-=-=
Content-Type: text/plain

I really enjoyed looking at that transcription.  Your transcriptionist
is great.
```

Next is the first file attachment, which is the `.docx` original. Here are the headers for it:

```
Content-Type: application/msword
Content-Disposition: attachment;
 filename="Episode 146 Gaurav Venkataraman discusses memory in RNA and DNA.doc"
Content-Transfer-Encoding: base64
```

All good; that's what it looked like before. Now if we scroll down, past all the Base64 data for the document, we can see the MIME headers for the converted PDF-A copy that Attachment Converter created:

```
Content-Type: application/pdf
Content-Disposition: attachment;
        filename="Episode 146 Gaurav Venkataraman discusses memory in RNA and DNA_CONVERTED1694624784.pdf
X-Attachment-Converter: converted;
        source-type=application/msword;
        target-type=application/pdf;
        time-stamp=1694624784;
        conversion-id=soffice-doc-to-pdfa;
        original-file-hash=305817139
```

We can see here that Attachment Converter added a special header called `X-Attachment-Converter` featuring information about the conversion that took place. What is says is:

- the original attachment was a `.doc` file
- it got converted to a PDF-A
- when it was converted (in a special numerical format)
- the name of this conversion in the configuration file
- here is number identifying the data from the original file, for debugging purposes

We can tell from the fact that the header `X-Attachment-Converter` starts with `X` that it is a custom header our application put into the email. Email software knows it can safely ignore any header that starts with the letter `X`, but it can also be set up to pay attention to the information in there, if it wants to do something special when it sees a header with a certain name. In our case, we put this information there both because Attachment Converter itself needs it, and because we would like future researchers to have a record of where converted attachments come from in the email containing them.

Scrolling down to the next attachment in this MIME multipart, we see the headers for the conversion to plaintext:

```
Content-Transfer-Encoding: base64
Content-Type: text/plain
Content-Disposition: attachment;
        filename="Episode 146 Gaurav Venkataraman discusses memory in RNA and DNA_CONVERTED1694624788.txt
X-Attachment-Converter: converted;
        source-type=application/msword;
        target-type=text/plain;
        time-stamp=1694624788;
        conversion-id=soffice-doc-to-txt;
        original-file-hash=305817139
```

We can tell we're at the end of the first email in the MBOX, first because there is a MIME delimiter with an additional two hyphens `--` after it:

```
--=-=-=--
```

The other reason we know we're at the end of the first email in the mailbox is that the next line is a From line, i.e. the beginning of the next email:

```
From keith@lib.uchicago.edu Wed Sep 13 16:25:31 2023
```

Remember, though this From line contains sender and timestamp information, that is for human eyes only; the only part your email software pays attention to is the word From, plus the space after, to tell that this is the beginning of a new email.

### 5.5 Advanced Configuration

Attachment Converter calls out to freely available external utilities to perform file conversions. The external utilities it uses by default are:

- `pandoc`
- `libreoffice`
- `pdftotext`
- `vips`

The conversions it performs by default are:

- PDF → plaintext
- DOC → plaintext
- DOC → PDF-A-1b
- DOCX → plaintext
- DOCX → PDF-A-1b
- XLS → TSV
- GIF → TIFF
- BMP → TIFF

- JPEG → TIFF

However, Attachment Converter also allows you to customize the conversions it performs with external utilities. You can set it up to use the utilities you already have installed to perform new conversions, you can modify one of the default conversion to use a different app of your choice, and you can also set it up to use a new app of your choice to convert a new type of attachment.

Setting Attachment Converter up with a custom configuration requires more computer expertise than using it with its default, out-of-the-box configuration. We won't get into the details of how to set up a new Attachment Converter configuration from scratch, but we will:

- customize the configuration in front of you now, so that you can get an intuitive feel for how much work is involved and what it generally looks like
- tell you who to contact at your university for help setting this up, if you think you might want to do it

### 5.6 Configuration File

With no special configuration, Attachment Converter performs the conversions described above on the input MBOX. However, if you create a configuration file, instead of performing the default conversions, Attachment Converter will perform the conversions the configuration file says to perform.

The easiest way to create a custom configuration is to go into the directory where you cloned the source code to and run our script for generating a configuration file:

```
$ cd ~/src/attachment-converter
$ ./make-config.sh > ~/.acrc
```

That second command will generate a configuration identical to what it does by default and put that configuration in a file called `.acrc` in your home directory. Opening that file up, it's a sequence of entries that look like this:

```
%source_type application/pdf
%target_type application/pdf
%shell_command /home/teichman/.config/attachment-converter/scripts/pdf2archive-wrapper.sh
%id pdf2archive-pdf-to-pdfa
```

Each entry has four fields: `source_type`, `target_type`, `shell_command`, and `id`. Their respective purposes are as follows:

- `source_type` is the name of the MIME type that Attachment Converter will flag for conversion, whenever it encounters this MIME type in the input mailbox
- `target_type` is the name of the MIME type that Attachment Converter will convert every attachment it finds that is of MIME type `source_type` to
- `shell command` is the path toward the UNIX shell script that Attachment Converter will use to invoke an external utility to perform a conversion
- `id` is the name that Attachment Converter will use to refer to this conversion: in this example, "convert all PDFs to PDF-A-s will be nicknamed `pdf2archive-pdf-to-pdfa`

That's all that needs to go into an individual entry. The configuration file is a list of these entries, each of which is separated by two line breaks.

If you've created a configuration file with this format called `.acrc` located in your home directory, Attachment Converter will find it there and use it instead of the default configuration.

### 5.7 Creating A Custom Conversion

We thought it would be fun to show you how to add ... still writing this part

### 5.8 TBD part

We provided this handout early so that if you wanted to follow the setup instructions before the workshop, you could. But we're still working on the final section on configuration in advance of the workshop.

During the workshop, you should be able to get a copy of the updated handout from the Box folder. If you've reached this point and would like to use this section on configuration for reference, we would encourage you to download the latest copy.

If you aren't sure whether the latest materials are up yet, please check our website for updates:

http://dldc.lib.uchicago.edu/open/attachment-converter