How to build your own dark archive (in your spare time)

A talk for the Cornell Digital Preservation Management Workshop
November 2004

Priscilla Caplan, FCLA

I'm going to talk about how we are building the FCLA Digital Archive. I hope to tell you about how we took what we thought we knew and tried to apply it, and how we found out what we really did or didn't know about digital archiving. I would have called this presentation "Moving theory into practice" but that title has already been taken.

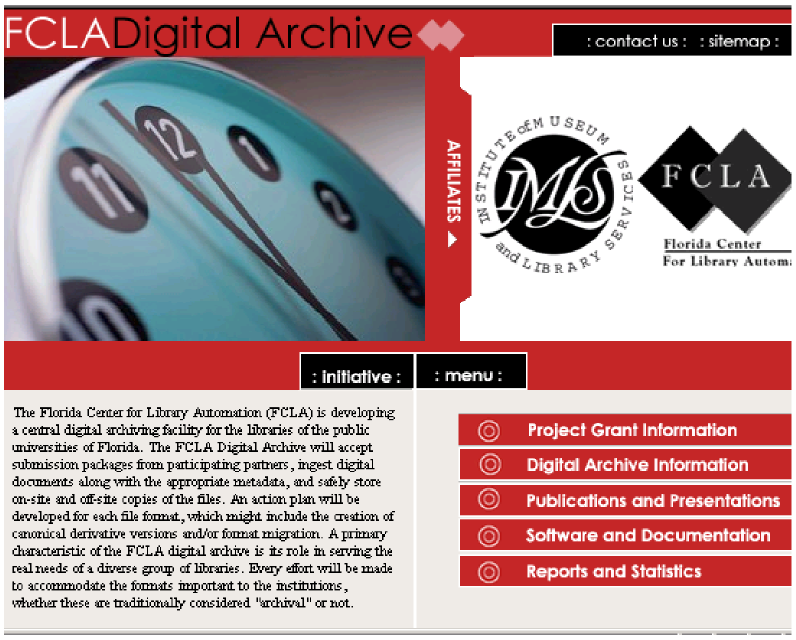
FCLA, the Florida Center for Library Automation, was established to provide centralized automation services for the essential needs of the libraries of the public universities of Florida. Until very recently, higher education in Florida was very centralized. There were ten public universities under a Board of Regents, and they shared central systems whenever possible. They all used the state personnel/payroll system, for example, and state purchasing systems, and the libraries all used the same integrated library system which was run by FCLA. A few years ago governor Jeb Bush reorganized state education by drawing up a new plan on a napkin, and now it's a little uncertain where things stand. But FCLA keeps going on as if there were still a State University System, and we try to encourage cooperation among the university libraries wherever possible.

When I came to FCLA in 1999 our main functions were running a central implementation of the integrated library system and coordinating consortial purchasing. I was told to develop digital library services and given a lot of rope. We established a program to support the libraries' retrospective digitization, called PALMM, the Publication of Archival, Library, and Museum Materials. We set up a program to support ETDs (electronic theses and dissertations). But it became clear fairly early that one infrastructure service the libraries really needed was a preservation repository for digital materials.

It makes sense to centralize that function. Most of the libraries aren't big enough to develop their own preservation repositories, and this isn't a function being developed by campus computing, which lags a little behind libraries and archives in awareness of this need. We felt we could develop a preservation repository as an option for the libraries. If a library wanted to use the OCLC Digital Archive or some other commercial offering, or if it wanted to develop its own digital archive, that was fine. But FCLA thought we could offer something at least as good as any commercial offering, and more cost-effectively. So we decided to build a digital preservation repository, and we applied for an IMLS grant to partially support that development, and we got the grant, for which we are very grateful.

I think of the process of developing the FCLA Digital Archive as a kind of journey, and of this presentation as a kind of travelogue. I'm going to start with some history, what we thought we were going to do compared with what we actually did. Then I'll go into some geography, describing the areas where theory meets practice, and some horticulture in pointing out some thorny details.

The FCLA Digital Archive


FCLADigital Archive
: contact us : : sitemap :
AFFILIATES
INSTITUTE of MUSEUM and LIBRARY SERVICES
IMLS
FCLA
Florida Center For Library Autom
: initiative :
: menu :
Project Grant Information
Digital Archive Information
Publications and Presentations
Software and Documentation
Reports and Statistics

The Florida Center for Library Automation (FCLA) is developing a central digital archiving facility for the libraries of the public universities of Florida. The FCLA Digital Archive will accept submission packages from participating partners, ingest digital documents along with the appropriate metadata, and safely store on-site and off-site copies of the files. An action plan will be developed for each file format, which might include the creation of canonical derivative versions and/or format migration. A primary characteristic of the FCLA digital archive is its role in serving the real needs of a diverse group of libraries. Every effort will be made to accommodate the formats important to the institutions, whether these are traditionally considered "archival" or not.

Our original idea, and the one thing that did not change over time, was the idea of building a dark archive. By "dark" I mean an archive with no real-time, online access to the content by anyone except repository staff. Dark archives are out of favor right now but we had some good reasons for it. We serve ten state universities and each of them has its own presentation systems and some have their own institutional repository systems. Some of the libraries use FCLA delivery applications but some use their own applications. Central Florida uses CONTENTdm, South Florida uses SiteSearch and Florida State uses DigiTool. At FCLA we don't have the software to replicate these access functions and we don't have any desire to; it would cost a great deal to acquire the software licenses, and it would take a huge amount of staff to support all these applications. So the idea of our offering presentation services on top of our stored repository content wasn't feasible.

It would have been possible to follow a model like Harvard's repository. They offer storage and repository services for content but make the content owners responsible for presentation services. The repository itself has no presentation interface, but it will deliver up content on request to presentation applications. However, in order to do that, we would have had to store service copies of files as well as preservation masters. We'd have to store streaming audio as well as master AIFF files, and JPEG derivatives as well as master TIFF files. Since we ultimately plan to do cost-recovery charging for the repository services, we didn't think our customers would want to pay for storing service copies. In sum, if we weren't going to offer presentation services, and if we weren't going to store service copies of files that could be fed into local presentation services, then we had to be a dark archive.

That decision vastly simplified many other aspects of design. For example, we don't have to deal with access rights and permissions, because we will only give content back to the owners who submitted it. We make them warrant that they have the right to grant us permission to conduct preservation functions, and we don't have to worry about anything else. It also makes physical security easier, since we don't have to worry about access over the Internet.

The decision to use tape as a storage medium followed from the dark archive. If you don't have to provide online access, tape becomes a reasonable option. We can get a terabyte of data on a tape that costs $250, which is substantially less than the same amount of spinning disk. It turns out that using tape has other complications and we may ultimately switch to disk, but the economics are such that we'd need a good reason.

We planned on a three year project with some IMLS funding. We're very grateful to the IMLS for taking a flyer on this project, and while they didn't pay for a whole lot – one staff position and some travel – having the IMLS grant gives us some legitimacy and visibility. In the original proposal, we were going to spend a year (nine months, actually) developing the archive and then spend the next two years collecting data and analyzing how to best do cost-recovery charging. At the time I thought that developing the software would be simple and that the economics was the really interesting part. As it turned out, we spent six months designing the software and around the time we should have started programming we threw out the first design and started all over again. We took another six months to do the redesign and model it formally in UML (universal modeling language). Now after about six months of programming we're planning to come up this month in production. The only service we'll have in operation is Ingest, so I'm telling the library directors: we'll be able to take your stuff, but we don't have Dissemination yet so you won't be able to get it back, and we don't have Reporting so we won't be able to tell you about it. But we estimate Ingest is approximately 60% of the code, and the other services will follow over the next year.

Treatment of any format is based on an action plan. The action plan specifies what to do with items in that format in the immediate short term (for example, normalize on ingest) and in the longer term (for example, do a forward migration). Originally, our plan was to ingest only materials in formats for which we had an action plan. So, there might be some time when we only took TIFFs and XML, and then later we'd take PDF files, and still later AIFF files and so on. It didn't take long to drop that idea, because the first priority of the library directors was for us to archive their ETDs. The ETDs are the intellectual product of the university, they have no printed equivalent, and the libraries are responsible for them. Most of our universities require the main body of an ETD to be a PDF file, but they allow associated files in any number of different formats. We couldn't very well ingest part of an ETD and reject the rest. So we had to accept files of any format.

That in turn forced us to establish two levels of preservation services; full preservation for those formats with action plans, and bit-level preservation only for formats without them. In bit-level preservation, we'd ingest the file and keep it secure and intact, but we wouldn't do active preservation strategies such as normalization or forward migration. If we later developed an action plan for the format, then we'd go back and re-ingest the object. I'll explain a little more about that architecture later.

Initially we planned to implement two preservation strategies – canonicalization and forward format migration. Canonicalization means creating a version of an object in a canonical or standard form. You don't see a whole lot about canonicalization in the preservation literature and there's a good reason for that – there aren't many canonical formats. We do a "softer" version of canonicalization that we call normalization. Whenever possible we create versions of files in normalized, hopefully more perceivable formats. For example, if we get a PDF file, we'll create page image TIFFs for every page in the PDF. The TIFF isn't a successor to the PDF, it likely loses some of the functionality of the PDF, but we're pretty sure we can keep TIFF files usable over the long term. Normalization is a form of hedging our bets. We'll try to preserve the PDF and do forward migration on it for as long as possible, but if that path finally ends and there's nothing more we can do, then we'll still have the TIFFs and their successor versions.

Finally, we planned to make our normalization and migration tools available to everyone as open source software. The way we envisioned it initially, we'd ingest materials into the repository, and at some point a forward migration would be required, and we'd write a program to do the migration then it available to everyone. However in our ultimate design, these migration tools are tightly integrated with the whole repository system software application, which we call DAITSS (Dark Archive in the Sunshine State). In DAITSS there are data file objects which know all about themselves. A PDF file object, for example, knows how to validate itself, extract its own technical metadata, and normalize itself into TIFF page images. So now instead of planning to make migration tools openly available, we're planning to make the entire DAITSS application available. Right now we're looking for a small number of partners to implement DAITSS at their own sites and help us finish the remaining services in a very controlled way. Then we'll release the entire application under some kind of open source license.

I'd like now to focus on a few areas where theory bumped into practice. I hope very much that the theory part will be repeating what you've already heard earlier in this workshop. We won't call that redundant, we'll call it reinforcement.

Theory 1: Preservation strategies

The first area is preservation strategies. You often read in the literature that there are two basic preservation strategies, migration and emulation. In fact, there are lots of different strategies, and many are used in combination. Ken Thibodeau from the National Archives and Records Administration has a marvelous slide that shows thirteen different preservation strategies plotted against a continuum with "preserve technology" at one end and "preserve objects" at the other. I love that slide because it really whacks you over the head with the fact that migration and emulation are only two of many strategies. In reality it's even more complex, because there can be several variations of any single strategy.

Format migration is a case in point – it has a number of variants including mass migration and migration on request. With mass migration, if format A is in danger of becoming obsolete, all files in a repository in format A are converted to format B. Later on, when format B threatens to become obsolete, all files of format B are converted to format C. There are two disadvantages to mass migration. The first is that it may take a lot of time and computing power to reformat large numbers of files. The second is that any format transformation bears the risk of introducing artifacts that weren't in the original, or losing some quality or functionality of the original. The more transformations you do and the further from the original you get, the more intrusive those changes can be.

This led some bright people at the University of Leeds to propose a different method, migration on request. (You know they weren't American, of it would have been called "migration on demand.") You can think of mass migration as a "just in case" strategy, and migration on request as "just in time." When format A threatens to become obsolete you write a program to convert from A to B, but you don't actually do the conversion unless someone asks for a file. Then at the point of delivery you convert A to B, or later on, A to C. This spares you the effort of mass migration but more importantly, it removes the risk of deterioration over successive transformations. You always keep the original format A and convert directly from A to whatever is current.

A downside of migration on request is that more conversion programs can be required over time. That can be ameliorated by yet another technique, a way of creating conversion programs so they can handle classes of similar formats.

When we were discussing the relative merits of mass migration and migration on request, we realized that regardless of the migration strategy we used, we would always want to keep the original file as it was submitted. Bit level preservation of originals seems to be emerging as a Best Practice in the digital archiving community. One reason is it aids in demonstrating the authenticity of the objects in the archive. The integrity of the original file can be demonstrated by comparison of message digests, and any transformation done to the original can be repeated if necessary. Another reason is that there is always the possibility that someone will come up with a better migration algorithm than the one that you used; if you always retain the original file, you can always re-do a migration.

Once we decided to retain the originals, we realized that the difference between mass migration and migration on request is tactical more than strategic. We could obviate the deterioration in multiple transformations by always working from the original, and we choose whether to do mass migration or migration on demand on a case-by-case basis.

In point of fact, our "hedge your bets" strategy has us doing forward migration not only on the originally submitted files, but also on the normalized versions of those files that DAITSS creates on ingest. So, for example, if we ingest a PDF file, we will try to migrate that file forward through as many versions of PDF and its successor formats as possible. But we will also normalize the PDF into page-image TIFF files, and we'll migrate those TIFFs forward as well. The hope is that if one line reaches a digital dead end, the other line may still be viable.

Theory 2: OAIS

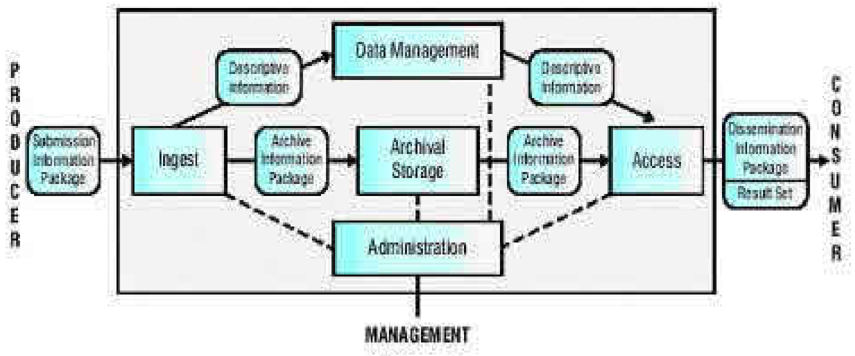


Figure 3

It isn't easy to find a preservation repository that doesn't claim to be compliant with the OAIS reference model. This is a bit of a bugaboo of mine because I haven't really seen too many OAIS-compliant applications. Part of the problem may be that OAIS itself doesn't provide much help in defining compliance. The specification requires a "conforming OAIS archive" to "support the model of information described in [section] 2.2" and to "fulfill the responsibilities listed in [section] 3.1."

For those of you who don't have OAIS memorized yet, the information model in 2.2 divides information into "content information" and "preservation descriptive information." Content information is a hybrid of data and metadata; that is, it includes the content data object itself, the *ding an sich* as it were, and what they call "representation information," which is anything necessary to make the content data object understandable to the designated community. Representation information in turn is divided into semantic information and structural information.

A file of scientific data, for example, might consist of a series of paired 32-bit fields, where the first 32 bits is interpreted as a time stamp and the second 32 bits is a floating point integer. That's structural representation information. The knowledge that the integer is a temperature reading at a particular place, and the time stamp is when the reading was taken is semantic representation information.

Remember that OAIS was developed by the Consultative Committee on Space Data Systems. This model of content information probably makes a lot of sense for science and social science datasets, but its harder to apply to textual or visual cultural heritage information, especially the distinction between semantic and structural representation information. In fact, the OCLC/RLG Working Group on Preservation Metadata in their analysis of OAIS decided that distinction was too hard to make and eliminated it from their version of the OAIS model.

Preservation descriptive information is a little more straightforward. This has four sub-categories: reference, context, provenance and fixity. Reference information is what we would call descriptive metadata – it identifies and describes an object. Context information is how an object relates to other objects, presumably including copies and other versions made by the preservation repository. Provenance documents the history of the content data object. Fixity information is actually misdefined in OAIS as authenticity but they actually do mean integrity mechanisms such as message digests. All in all this isn't a bad model for preservation metadata although it is a bit object-centric.

There are six responsibilities a conforming OAIS archive must fulfill:

> 1. Negotiate for and accept appropriate information from information Producers.
> 2. Obtain sufficient control of the information provided to the level needed to ensure Long-Term Preservation.
> 3. Determine, either by itself or in conjunction with other parties, which communities should become the Designated Community and, therefore, should be able to understand the information provided.

4. Ensure that the information to be preserved is independently understandable to the Designated Community. In other words, the community should be able to understand the information without needing the assistance of the experts who produced the information.
5. Follow documented policies and procedures which ensure that the information is preserved against all reasonable contingencies, and which enable the information to be disseminated as authenticated copies of the original, or as traceable to the original.
6. Make the preserved information available to the Designated Community.

Point four is obviously closely related to the information model. You need to preserve with the object whatever is necessary to interpret the object, and this is iterative, until you finally reach a level that is independently understandable to your designated community. To interpret the time and temperature dataset you need the code book. If the code book is in English and your community is English-speaking, you may be done. If the code book is in Esperanto, you may need to include an Esperanto grammar and dictionary.

With DAITSS we tried to take a hard line interpretation of OAIS. We do have formal signed agreements with our "producers," the ten libraries of the state university system. To control the information to the level needed for long-term preservation, we have stringently documented SIP and AIP contents, and we record all of the prescribed metadata in the system management tables. We also bundle a complete set of the metadata including representation information and preservation descriptive information and store it redundantly with the content data object in the AIP. By policy, all of our representation information has to end in a specification, preferably on paper. So, for example, if we have a TIFF image, we not only record the technical metadata about the particular TIFF, but we also have to have a copy of the TIFF specification. Our designated community is the same as our producer community, the participating libraries.

We also rigorously follow the separation of functions described in OAIS 4.1. When I give presentations about the DAITSS architecture, I usually use the OAIS diagrams to illustrate it, because they describe DAITSS as well as OAIS. Producers have to submit a SIP with a METS format SIP descriptor to the DAITSS Ingest service. The Ingest service talks to Data Management to update the system management tables and to Archival Storage to write the AIP to storage. To access something, the request goes to the Dissemination service (which is our name for the OAIS Access function) which assembles a DIP with metadata about the object from Data Management and the object itself from Archival Storage.

If you look at the model closely, you see there is no way to alter an AIP in Archival Storage. It can't happen in the Archival Storage function, which only writes what Ingest gives it, and it can't happen in Data Management, which is essentially a reporting function. The only place an archived object could be changed is in Ingest or in Access. In DAITSS, all reformatting happens in Ingest, whether that be normalization or forward migration. If a file is archived in format A and later needs to be migrated to format B, the only way to do this is to disseminate the object via the Dissemination Service and re-ingest the object via the Ingest Service. As part of Ingest DAITSS will determine the file needs to be migrated and will create a migrated version at that time. Although that may sound un-intuitive, it turns out to be a very simple and elegant architecture that greatly simplifies what would otherwise be some very complex processing.

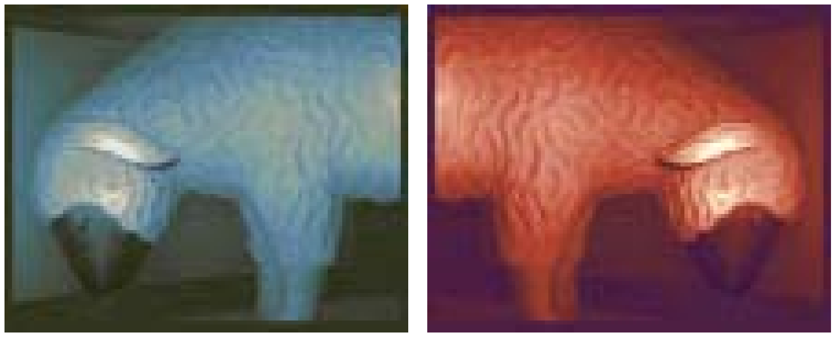
It may be worth talking about Ingest in more detail, since this is where most of the processing in DAITSS takes place. A library must create a SIP for each intellectual entity it wants to archive. Our specifications require the SIP to contain a minimal amount of metadata in the form of a METS document and all the files needed to render a version of that intellectual entity. (What we mean by that is, if you're sending us a page-image book, for example, you need to include all the pages in the package.) The libraries FTP the SIP to a particular directory that the DAITSS system uses as input.

When DAITSS finds a new package, the first thing it does is validate the SIP descriptor to make sure it's proper METS and adequately describes the package. It extracts any metadata in the SIP descriptor and saves it for later. Then it looks at every file in the package. First it tries to identify the file format using clues like the filetype extension and by parsing the file. Once DAITSS knows what type of file it is, it creates a file object that is really pretty smart. DAITSS tells the file object to validate itself, and to extract its own technical metadata. Metadata supplied in the SIP descriptor is compared with the extracted metadata and any conflicts are resolved.

If any external files are referred to that are not included in the SIP, DAITSS attempts to harvest them. For example, the METS descriptor will refer to the METS schema, which might not be included in the SIP, so DAITSS will try to find a copy of the METS schema and download it into the archive. (Actually, in the case of very commonly used files like the METS schema, DAITSS creates what we call a "global file" that is only stored once no matter how many times it is used.) Once all the necessary files have been obtained, a new SIP descriptor is created that describes the new augmented SIP.

The Ingest service looks to see if any of the files need to be forward migrated, and if so it tells the file object to migrate itself. Then we see if any file needs to be normalized and the file object is told to normalize itself. Metadata is created describing all the files, whether submitted or created by DAITSS, and every relationship between files, and every event that has happened to every file, and the system management tables are updated with that metadata. Finally an AIP is created which consists of the augmented SIP, all migrated and normalized versions of files, and a copy of all of the metadata that was added to the management tables in a local XML schema. The AIP is passed from Ingest to the storage interface, and Ingest looks for the next package.

*Theory 3: Risk Management **



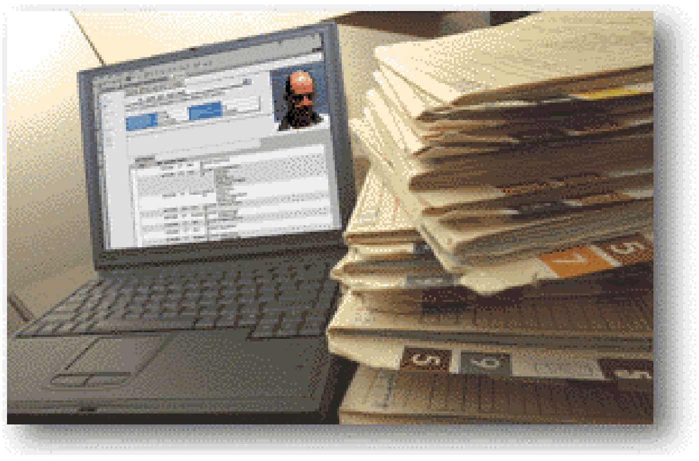
Usually when we think of risk management in digital preservation we think of formats. You need to evaluate the risk of format obsolescence and weigh that against the risk of degrading the file in migration. For the FCLA Digital Archive, we create an action plan for each format saying whether to normalize that format during Ingest, what actions to take in the short term, what the long-term preservation strategies are, and when to review the action plan. Action plans in turn are based on fairly in-depth analyses of the formats that we publish in background reports.

The background reports contain information such as a description of the format, a pointer to the specification(s), and how a parser would recognize the format. We look at the history of the format, particularly in terms of the longevity of each version, and they look at the "openness" of the format: whether the specification is available, whether the format is an official standard, how it is maintained. To get a feel for whether there are likely to be open source or commercial migrators, we look at platform support and perceived popularity. We look at legal issues, such as whether the format uses patented technologies. It's a lot of work, and we look forward to the day when there are central format registries that contain much of this information. In the meantime we've made our own background reports and action plans available on our website at www.fcla.edu/digitalArchive/ for anyone to use.

But risk management goes beyond format analysis. Our fundamental approach to reducing risk is redundancy. For redundancy of content we write three separate independent masters, two local and one remote. Each master is independently backed up with local and remote backup copies. We routinely create normalized versions, and as I mentioned earlier we always do bit preservation of the original. If we do multiple migrations on a format (that is, format A to format B to format C) DAITSS by default will keep the intermediate versions, although these can be deleted by repository management. We are concerned that all this redundancy of content might get expensive over the long term, and we might have to give up some of it, but right now we feel that we have so little experience with digital preservation that it is better to err on the side of caution.

Redundancy is built into other areas as well. We calculate two message digests on every file, MD5 and SHA-1. We store metadata in relational tables in the repository management database and redundantly in XML in the AIP. And so on.

Theory 4: File formats



I know you've spent a lot of time in this workshop looking at file formats in terms of their preservation qualities. It's fairly well accepted, I think, that a good preservation format has to pass the "fidelity test" (is it true to the original, or if born-digital, does it replicate the original experience?) and the "future test" (is it open, well documented, well supported?). You prefer formats that don't embed proprietary technologies, and that don't have access inhibitors such as encryption.

By policy, the FCLA Digital Archive will accept files of any format, although we can only do full preservation on formats for which we have action plans. However, we still encourage the use of preferred formats. We've published a list of recommended file formats on the archive website which we update annually, and we work with the graduate schools at the state university to help them encourage ETD submissions in preferred formats. DAITSS will also normalize a file on ingest from a non-preferred to a preferred format when possible.

However, we've found that file formats are much less straightforward than you'd think from most discussions, starting with the question, what's a file format anyway? For one thing, many formats support profiles. For example, the Adobe TIFF specification describes four baseline provides and several extension profiles, and there are many non-Adobe TIFF profiles some of which (like GeoTIFF) are completely compatible with TIFF 6.0 and others (like RichTIFF) are not. For most purposes you really need to know the profile as well as the format.

For another thing, most of the technical characteristics you need to know for preservation purposes adhere to internal bitstreams rather than files. A TIFF file can contain an arbitrary number of embedded images, each with different technical characteristics such as colorspace. In the DAITSS data model, each file object is composed of zero or more bitstream objects and most of the technical metadata is recorded at the bitstream level.

Moreover, files can be encapsulated, encrypted, compressed, etc. such that file formats are often layered. A thesis consisting of a PDF and an AVI file could be tarred such that instead of two files with type PDF and AVI you have one file of type TAR. If the TAR file is compressed by gzip then you have a different file of type TGZ, which you could then encrypt creating yet another file. I call these formats layered because to get back to the original two files you have to undo your transformations in order: unencrypt the TGZ, then uncompress and untar it.

Perhaps the most difficult aspect related to file format information for preservation is the specification of environment information. Environment is what we call the hardware and software platforms on which a file can be rendered or used. For very long term preservation you can't take this for granted. A complete software environment is multi-faceted and can include not only the rendering application, but the runtime environment, operating system, drivers and other components. Each component can also have dependencies on other modules. Andrea Goethals, our format specialist at FCLA, wrote out as an exercise all the dependencies for a single rendering program, xpdf, and found it required 44 different external library modules. Specifying a single software environment for a file format can take some work. The same is true for hardware.

Of course, most file formats can be rendered in multiple environments. Do you record them all, or some subset? Ought you to determine the optimal environment, or the minimal workable environment? These are problems repositories are grappling with. I know one repository that has decided to record two environments, one that is common and known to work, and one that is recommended by the repository's criteria. Another preservation repository is taking pains to record three or four different environments for each format.

It would be lovely if someday we had an environments registry similar to file formats registries.

So, to conclude, I know you'll forget all this detail, but I hope you'll take away these two points. First, you really do need to know the theory, it really is important. Workshops like this are terribly important. But remember that the theory only takes you so far, and when you start designing or implementing systems, you get have an interesting geography lesson. And second, I think digital preservation is possible. We don't quite know how to do it yet, but with a good grasp of theory and sufficient determination, we'll get there. I think we're going to see a whole crop of second generation repository systems like DAITSS that are designed from the start to deal with the complexities of digital preservation. We're going to see the development of format registries and environments registries and other tools to implementing preservation easier. I'm looking forward to seeing what happens. That's why I'm ending this talk with a shot of our DAITSS logo. The future's so bright, we gotta wear shades.

DAITSS
Dark Archive in the Sunshine State

* Images taken from Terry Gips, Dolly's Dilemma, (http://www.coldmeadow.com/tgweb/tgart/dolly/ddindex.html) and used by permission of the artist.