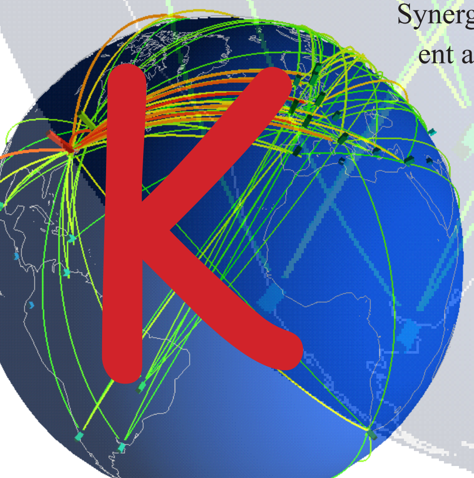
K is for Knowledge. In efforts to spread it.
Cornell is digitally, of course, at the head of it.
Database arXiv represents the ideal of it.
Cornell faculty get credit for much of it.

You may know that scientists are putting more and more research articles on the Web instead of submitting them to traditional journals. You may know that the major database for this, arXiv, which is free and is minimally moderated, contains over 380,000 articles in physics, math, and CS and gets 50,000 new submissions each year. But do you know Cornell's role in these ventures?

ArXiv is maintained by the Cornell Library. It is the brainchild of Cornell Physics-CIS prof Paul Ginsparg. He originally developed it while at the Los Alamos National Lab and still does research on it —he got a Macarthur fellowship for it. CoRR, the CS part of arXiv, is due to CS prof Joe Halpern, who pushed the ACM into supporting it and worked with Ginsparg and CS researcher Carl Lagoze to incorporate it into arXiv.

Many of the behind-the-scenes software ideas in today's digital libraries are due to Cornell people like CS researchers Dean Krafft, Jim Davis, and Lagoze. In the 1990's, they worked with others including CS prof Bill Arms —then at CMU— on a DARPA project to put tech reports on the web. Lagoze and colleagues built the Dienst architecture, which led to the Fedora project and the widely used metadata harvesting protocol OAI-PMH. Today, this technology underlies the NSF's National Science Digital Library, the state of the art for web-based digital libraries, which has its technical leadership here at Cornell.

Synergistic things happen over the years when quality researchers in different areas talk to each other.



L is for Languages, compilers, and such.
A field we're involved in so very much.
Why? —naive people may suddenly ask;
'Cause notation's oft key to solutions of tasks.
As Benjamin Whorf once said he did find,
Language does shape the thought and the mind.

We have been heavily involved in languages and compilers from the start! The PL/C compiler in the early 1970s; the first text on compiler construction; Tim Teitelbaum's Cornell Program Synthesizer; the language Russell, semantics (e.g. interference freedom, the basis for proving parallel program correct, was developed here) —these were influential projects in the 1970s and early 1980s.

Compilers themselves have remained an area of intense work here. For example, Keshav Pingali has had a long an influential project on compiling for parallel computers, which has expanded to deal with many more issues in compiling. Radu Rugina also works on compiling and program analysis to make software more robust, secure, and efficient.

Much of the current language work is inspired by security issues. Besides the language-design work in Andrew Meyers group, there is emphasis on analysis and synthesis methods that provide mechanical means for ensuring that a program and its execution will satisfy certain properties —a field called "language-based security". Dexter Kozen works on various aspects of proof-carrying code, and Fred Schneider, chief scientist on the multi-university NSF grant *TRUST*, and his students pioneered the use of in-lined reference monitors to check for violations of security policies.

Like other work in CS at Cornell, this work is tied to principles and often addresses problems that transcend technology or specific engineering issues. "Think first, build second" is a succinct characterization of our primary modus operandi. Perhaps that is why Cornell has been a leader in this field.