SSI, better known as Server-Side Includes

"Server Side Includes" (SSI's) are commands that you give to your webserver by placing them right inside your HTML code! You can include a standard copyright disclaimer at the bottom of every page of your website, and if you need to change it, you only have to change it once!

When a browser requests a file from your webserver, the server can be made to look for Server Side Includes within HTML files and execute them. The SSI programs provided below are programs which are run via the "exec" SSI command. The SSI command within your HTML code will be replaced by the results of the SSI program. (Even if one uses the "View Source" function of their web browser, the SSI will look like it was actually part of the page!)

SSI's are very powerful. They're better than Java for the types of things they do, since not everyone can use Java - but anyone capable of downloading an HTML page will see the results of a Server Side Include!

The types of things you can do with Server Side Includes range from simple counters and "current time" displays on your page, to rotating banners and the ability to show different content depending on the type of browser someone's using, or where they came from (the page that got them to your site)!!!

Server-side includes allow you to create documents that include information that is re-generated each time the document is accessed.

With server-side includes, you can:

Include a number of parameters in your document automatically, including:

- The file name of the current document.
- The URL of the current document.
- The last modification date and time of the current document.
- The current date and time.
- Include the last modification date and time of any arbitrary file on the system. This can be used to create advisories such as ``Be sure to check the "late-breaking news" file... it was last updated on (last update)."
- Include the size of any arbitrary file on the system. This can be used to create advisories such as ``Click here for a full-size image, but watch out, it's (size)."
- Include the text of another file, either from somewhere else on the system or from somewhere else in your Web server. For example, if you have some ``boilerplate" that you want to appear at the top of every document, you can put that in one file and then include that file in all your other documents.
- Include the output from a CGI program. This differs from referencing a CGI program in an anchor (between <A> and </A>) in that a CGI program executed via a server-side include is executed automatically when the document is accessed, as opposed to being executed only when the reader clicks on the anchor.
- Execute any arbitrary system command. You cannot capture the output from one of these commands (to do that, you have to make a CGI script ``wrapper" for the command), but there are a number of commands for which this is not necessary.

Before deciding to use server-side includes, you need to understand the performance issues involved in doing so. To summarize them in one short sentence, server-side includes make your documents load more slowly.

Under the default Purdue ECN setup, server-side includes are turned off. When a browser such as Mosaic or lynx requests a document from the server, httpd (the server program) opens the file and sends the contents back to the browser. When it does this, httpd does not try to attach any meaning to the contents of the file. In fact, it doesn't even look at it to make sure it's what you say it is. For example, if you were to accidentally place an image into a file called foo.html, httpd not only would not care, it wouldn't even know. It just opens the file, reads a bunch of bytes, and sends them back to the browser.

Server-side includes though, as their name implies, require that the server (httpd) do something. That means that when server-side includes are turned on, and a browser requests a document, httpd has to not only read the bytes from that file, it has to look at them and try to understand them. It doesn't have to understand everything, but it does have to search the entire document for the special symbols that indicate a server-side include. And, when it finds a server-side include, it has to do whatever is required to process that include and capture any text that it generates.

All this searching of documents takes time. In fact, it takes enough time that people accessing your documents will probably notice it. Thus, you don't want to just turn on server-side includes everywhere; people will probably start complaining that your server is too slow. Instead, you will want to carefully evaluate which documents need server-side includes, and turn server-side includes on only in those areas. Leave them off everywhere else (that's why they're off by default in the Purdue ECN configuration).

"Server Side Includes (SSI), also known as Server Parsed HTML (SPML), provides a convenient way of performing server-side processing on an HTML file before it is sent to the client. SSI provides a set of dynamic features, such as including the current time or the last modification date of the HTML file without developing a CGI/PERL program that performs this function. SSI can be considered as a server-side scripting language."

If this description sounds a too bit technical for you, I'll paraphrase. Server Side Includes are the simplest, easiest way to generate dynamic web sites without any technical expertise of any sort.

Server Side Includes use simple HTML style commands you can add to your web page to either import a text page or run a CGI/PERL script before the page is served to the client (viewer of the page). If you can imagine a web page with an HTML table in the center, composed of four cells, each separate cell presenting the client with a different web page, or an executed CGI/PERL script, you can begin to see the power of the Server Side Include.

Before we get into how to use SSIs in your pages, here are a few disclaimers. If you have access to an Apache style server, first ask your administrator if SSIs are enabled. They are disabled by default. Due to the differences in server configurations, the exact commands described here may not work on your server. Contact your Web Server administrator for the exact commands used on your server.

With that out of the way, here are actual HTML page samples incorporating SSI:

filename: ssihtml.shtml

```
<HTML>
<HEAD>
<TITLE>Server Side Includes HTML Sample</TITLE>
</HEAD>
```

Example of a server side include inserting a block of text onto a Web page:

<!--#include virtual="directory/insertpage1.html"-->

</BODY>
</HTML>

filename: ssicgi.html

<HTML>
<HEAD><
TITLE>Server Side Includes CGI Sample</TITLE>
</HEAD>
<BODY>

Example of a server side include inserted a script on a Web page:

<!--#exec cmd="cgi/myscript.cgi"-->

</BODY>
</HTML>

Server Side Includes can be tremendous time savers in large Web project. Consider the amount of time saved if every page on your 1000 page Web Site had the same footer and you needed to modify a graphic in that footer. Normally, you would have to go into each page to the do the modification (that is, if you are not using an elaborate HTML publishing program of course). Since you are now an expert in the use of SSI, you simply have to change one html file and all of the pages on your site are instantaneously changed for all to view (miraculous isn't it).

So you ask, "Why aren't more people using SSIs if it is so easy to set up?" Well, there are some caveats.

First, many web servers require you to name your pages with the .shtml extension. This causes great heartache among some purists.

There are some security issues involved in the exec cmd call, so some servers have script execution turned off.

Extensive use of SSIs can effect server performance.

Most folks just haven't tried them.

There are additional advantages to using SSIs. SSIs allow some fairly advanced functions, such as, browser detection, CGI environment variables display and more. For Apache Servers, there is even an Extended SSI module (XSSI). This SSI enhancement allows for if-then-else statements, user-defined variables and other commands similar to those used in CGI/PERL scripts.

With a little imagination, Server Side Includes give you the capability to create dynamic pages in little or no time, without any programming skills. Adding this weapon to your arsenal of design tricks may be all that's needed to beat out your competition in your next big project bid.

The commands that the server looks for are actually kept inside of comment lines. A comment line in an HTML document looks like this:

<!--This is a comment-->

The '<!--' and '-->' are what contains the comment. This will not be displayed by the browser that is viewing the page.

By using a special format inside of the comment, the server will recognize it as a command, rather than just something to skip over and ignore. A server-side-include looks something like this:

<!--#exec cgi="filename.cgi"-->

The important part is the '#' at the beginning of the comment, which tells the server that this is a command. This is followed by the keyword 'exec', which is a command telling the server to execute the following program. The cgi="filename.cgi" portion tells the server exactly which program to execute. #exec is just one of a few possible commands, which will be covered below. For now, we'll use this format as a specific example.

It's not quite that easy (but what is? :). Since server setups differ for different machines and sites. Some things to keep in mind:

Not all sites have SSI's enabled at all. You'll have to mail your sysadmin to see if you can use them.

Some sites do not allow scripts do be executed in user's home directories. This means that you cannot do any of this simply. Ask your sysadmin, once again.

Some sites have #exec disabled, but not other SSI's.

Some sites do not have Perl installed on their system.

If all else fails, mail your system administrator. Ask if SSI's are enabled, whether you can have scripts in your own directory, whether you can have access to the /cgi-bin directory (where most scripts are stored), whether you have perl or not, etc. If he does not know how to fix these things, tell him that a REAL sysadmin would know how.

(Source unknown)