# HP Software Transition Kit (STK)

Peter Swärd HP Software Engineer
Transition Engineering and Consulting
Peter.Sward@hp.com

# Agenda

- STK Overview
- abiscanner Overview

- Q&A

# What is STK?

The STK is a collection of **documentation** and **tools**, linked using hypertext, web-based technology to help developers get their software ported/transitioned from one version of:

HP-UX to a newer version of HP-UX

**It helps developers with questions such as:**

- Do I want a 32-bit or 64-bit version of my software?
- Can I qualify my software (for example, run a 10.x executable on HP-UX 11.x) or do I need to port (create a new HP-UX executable on my destination platform)?
- How can I transition my software to Itanium Processor Family (IPF), the new Intel and HP architecture?
- What new features does the latest HP-UX release offer, and how can I take advantage of them?

# STK Users

**STK Users:**

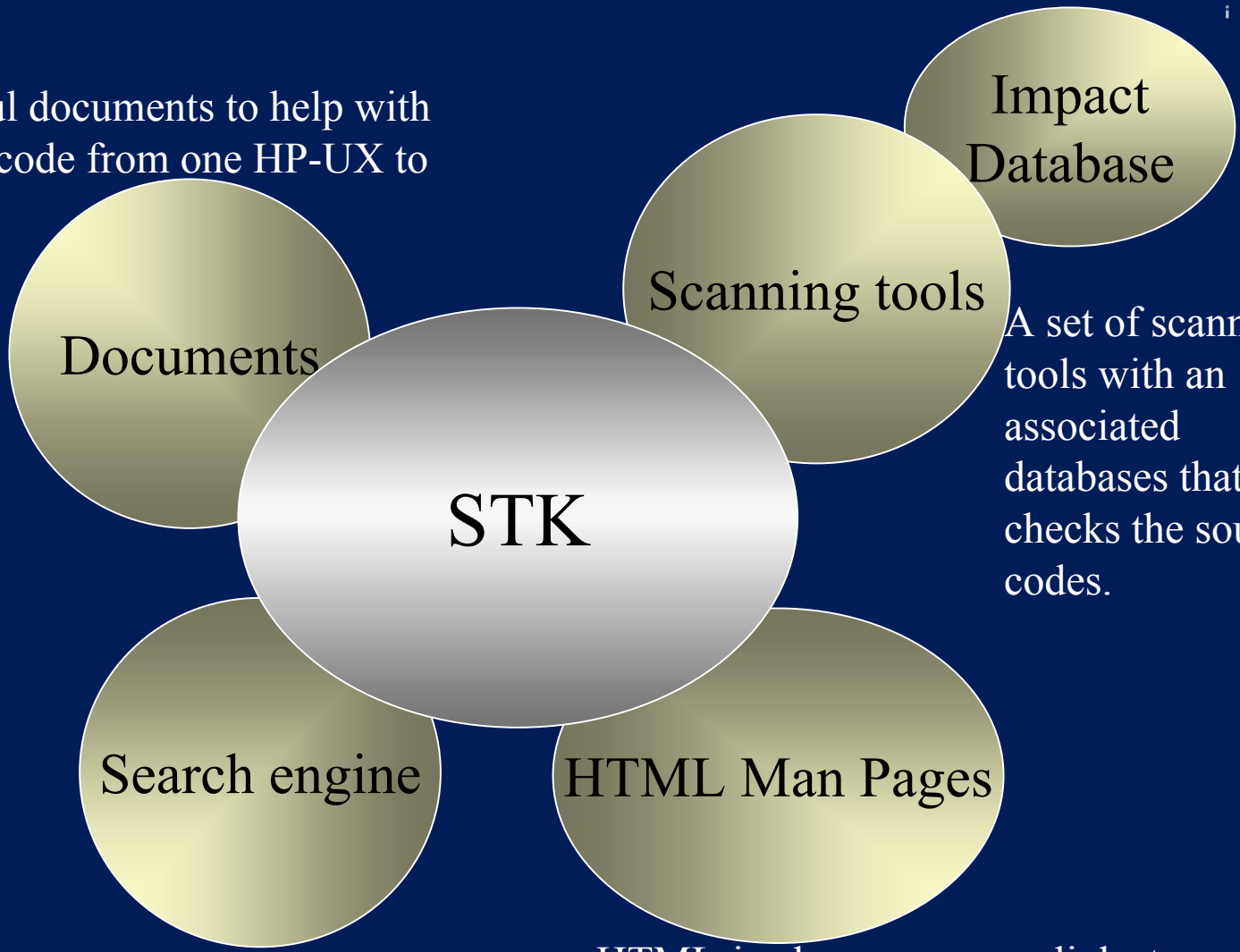
1. General software developers looking at HP-UX
2. HP partner developers porting to HP-UX
3. HP technical resources working with partners
4. HP consultants working with HP-UX and new systems
5. HP internal developers

# STK Coverage

1. HP-UX to HP-UX transitions
   A. HP-UX 10.x to 11.0
   B. HP-UX 11.0 to 11i
   C. HP-UX 11i to 11i v1.5
   D. HP-UX 11i v1.5 to HP-UX 11i v1.6
   E. HP-UX 11i v1.6 to HP-UX v2.0.

# What does STK contain?

A set of useful documents to help with transitioning code from one HP-UX to another.




A set of scanning tools with an associated databases that checks the source codes.

A search engine which searches the man pages, impact pages and documents.

HTML-ized man pages or links to HTML-ized man pages on docs.hp.com.

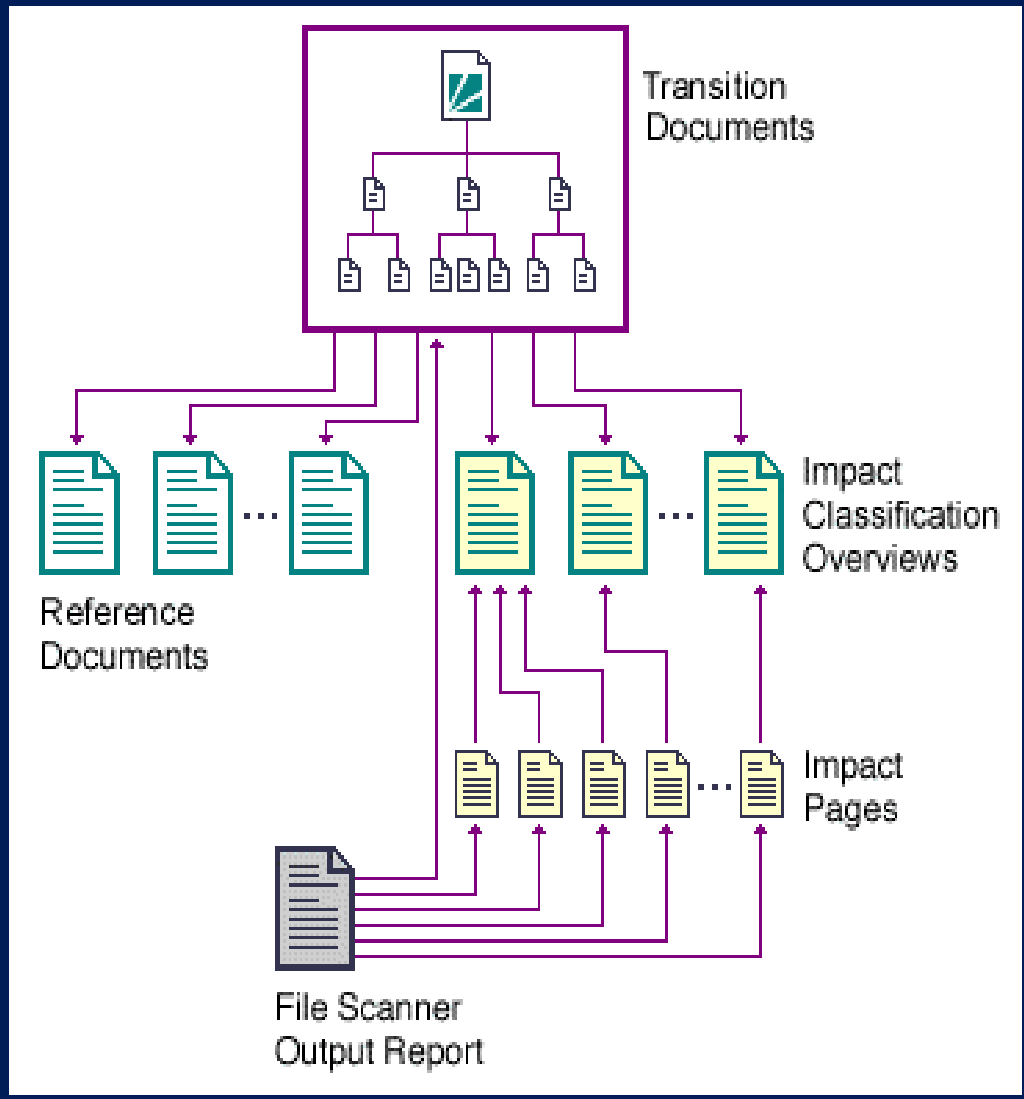
# STK Documentation

STK Documents

Mostly HTML based documents:

- Transition documents
- Reference documents
- Impact classification overview
- Impact pages
- File Scanner output report
- Partner documents

Also available in Japanese



# Transition Documents

Describe procedures people should use to investigate, plan and perform the transition of their software:

- ✓Source code transition
- ✓Qualifying source code
- ✓Porting to HP-UX destination platform
- ✓How to update systems
- ✓Using the filescanner tools
- ✓Transitioning C programs to the 64 bit model
- ✓HP-UX 64-bit compiler and linker changes
- ✓HP-UX 64 bit porting concepts
- ✓Understanding impacts

# Reference Documents

A library of useful reference documents

**64 -bit computing**

**ANSI C++ programming language**

**assembly language**

**binary compatibility**

**C programming language**

**COBOL programming language**

**file systems**

**Fortran programming language**

**HP -UX operating system**

**internationalization**

# Impact Documents

Describe the API impacts that can occur in transitioning source code to the next version of HP-UX

32 - 64 bit interoperability impacts summary


64 - bit API impacts  summary


binary compatibility impacts summary


date impacts summary


HP  aC++ and C compiler impacts summary


HP  Fortran compiler impacts summary


HP Micro Focus COBOL  4.x compiler impacts summary


IPF architecture impacts summary

# STK File Scanner

scansummary

- helps investigate or plan a transition
- reports **number and types** of API transition impacts in source files

scandetail

- helps perform a transition
- identifies **each instance** of an API transition impact in source files

# STK File Scanner

Scan C, C++, Fortran, COBOL, scripts and Makefiles

Scan for:

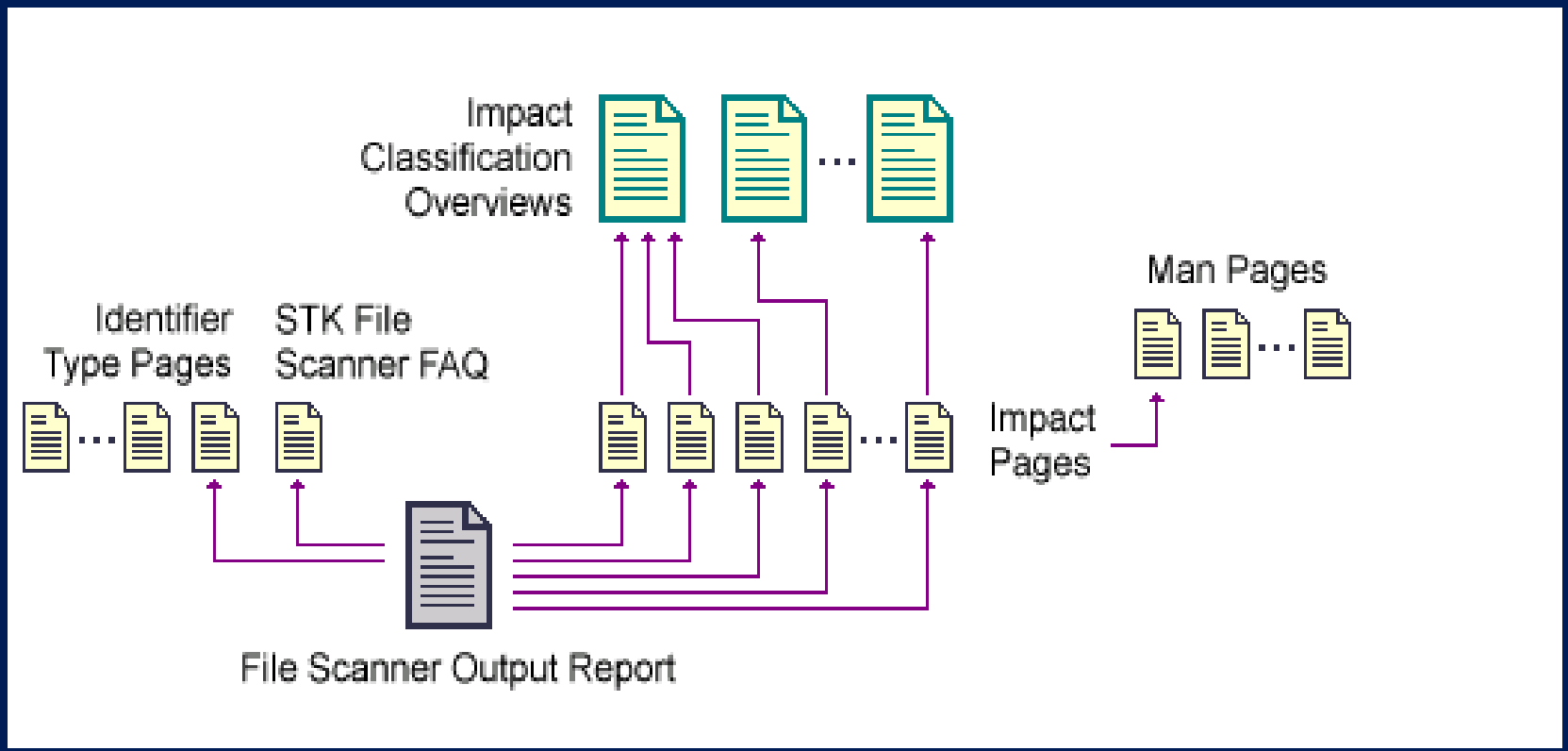
- functions
- commands
- macros
- structures and structure members
- header files
- language keywords
- libraries
- variables

Output formats

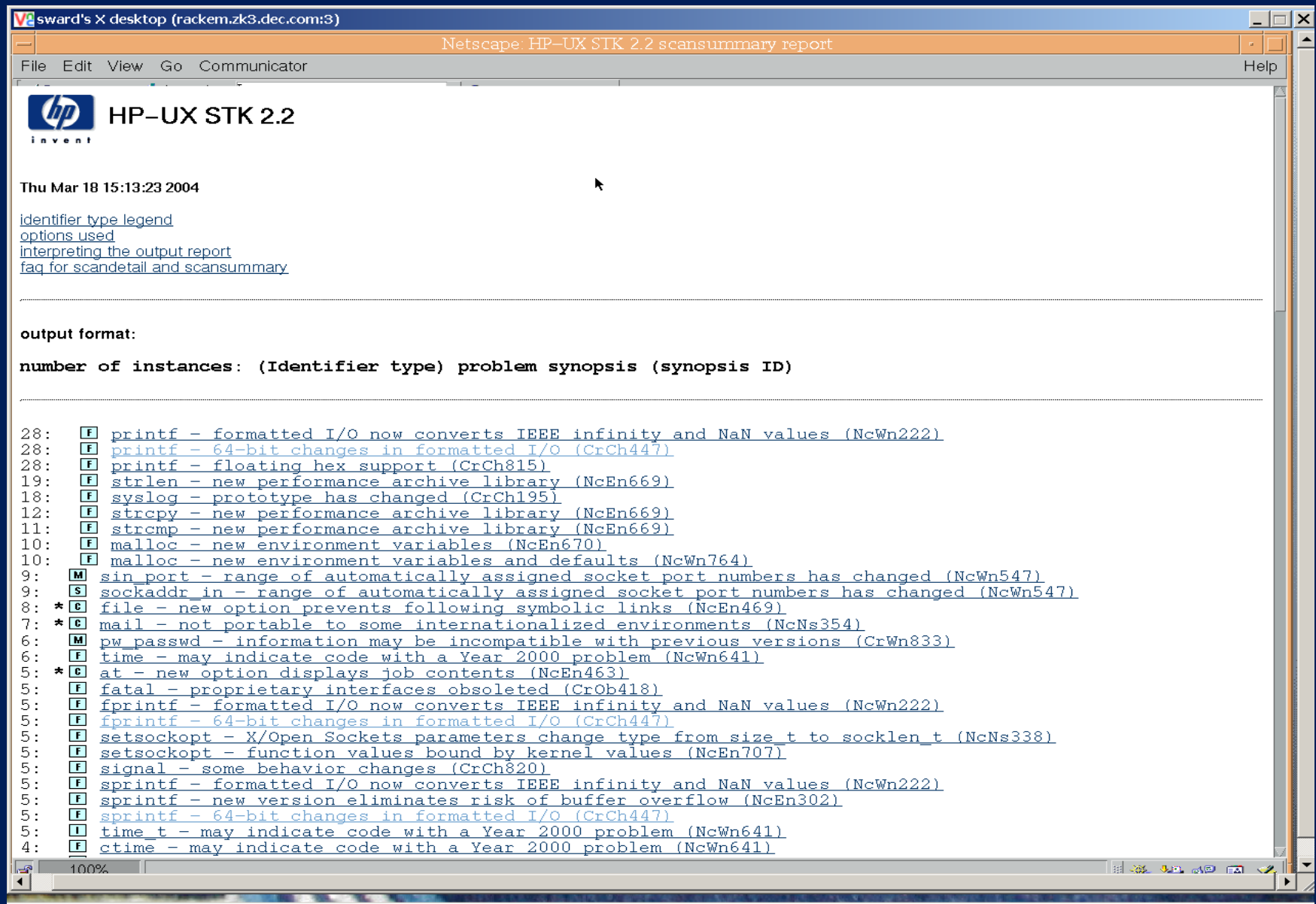
- html (default)
- text

# STK File Scanner Report

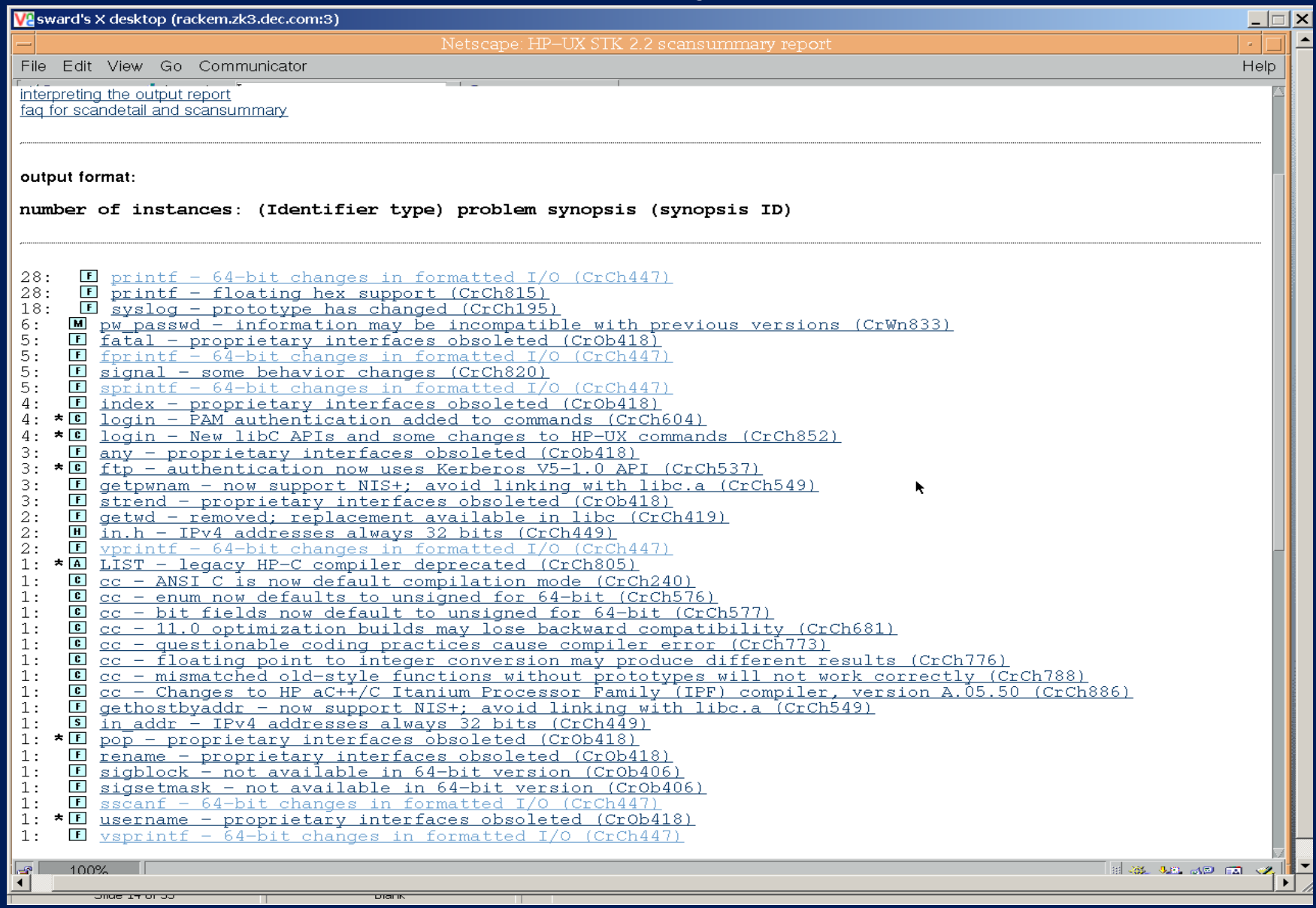
The file scanners generate reports that provide links to detailed impact pages describing each impact and its solution. The impact pages in turn link to impact classification overviews and man reference pages, which provide additional background information.



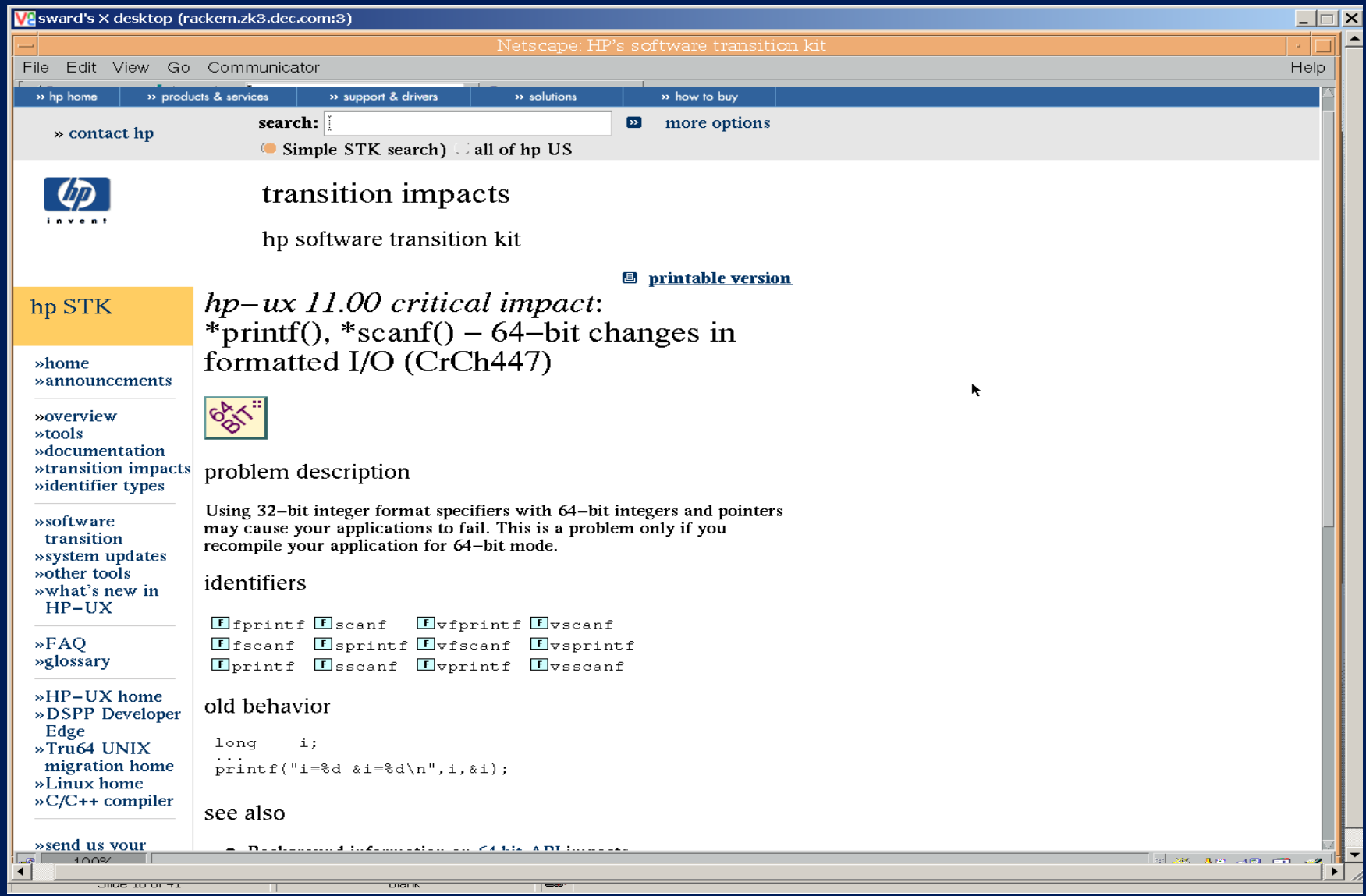
# Scansummary



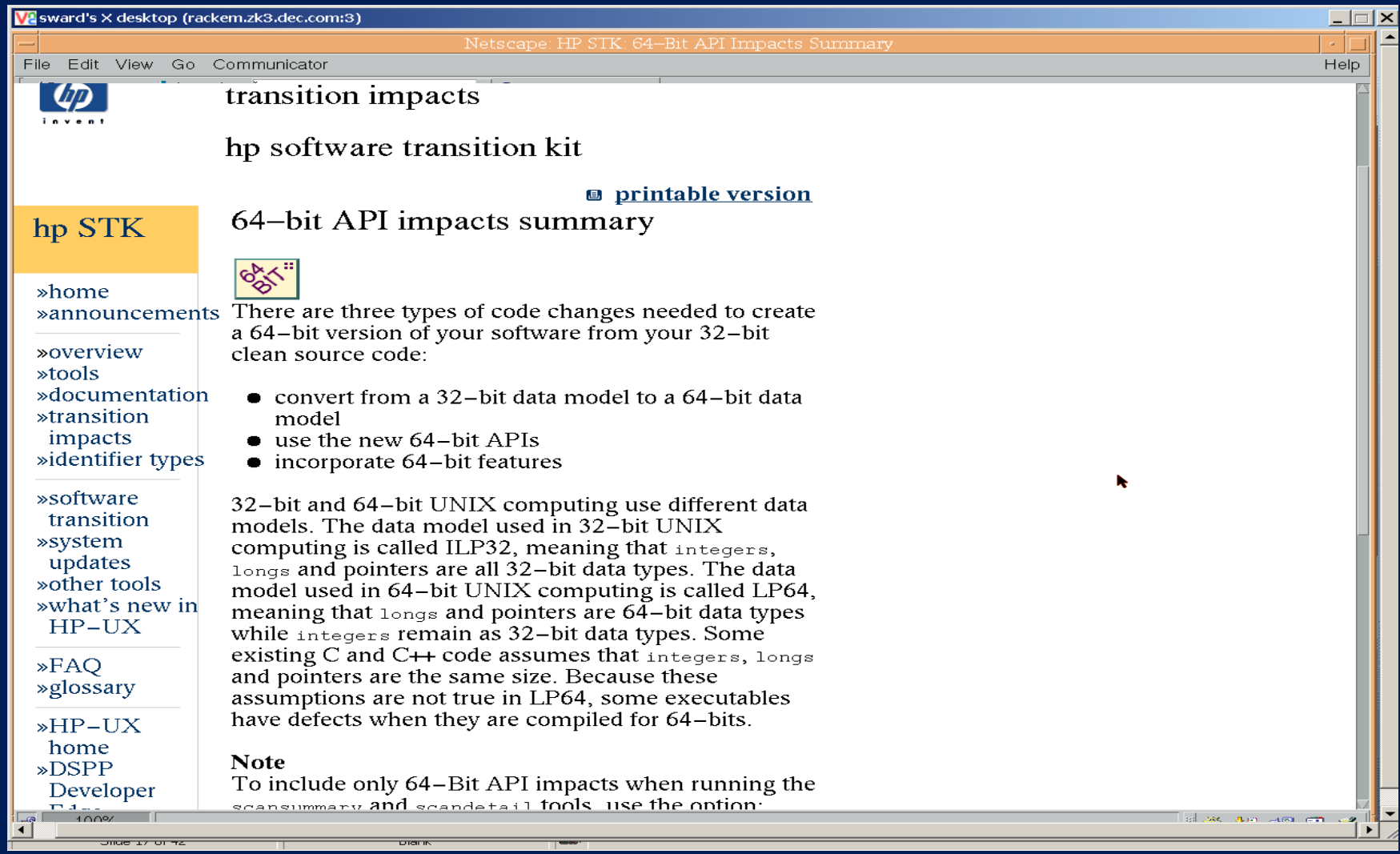
# Scansummary +S CR



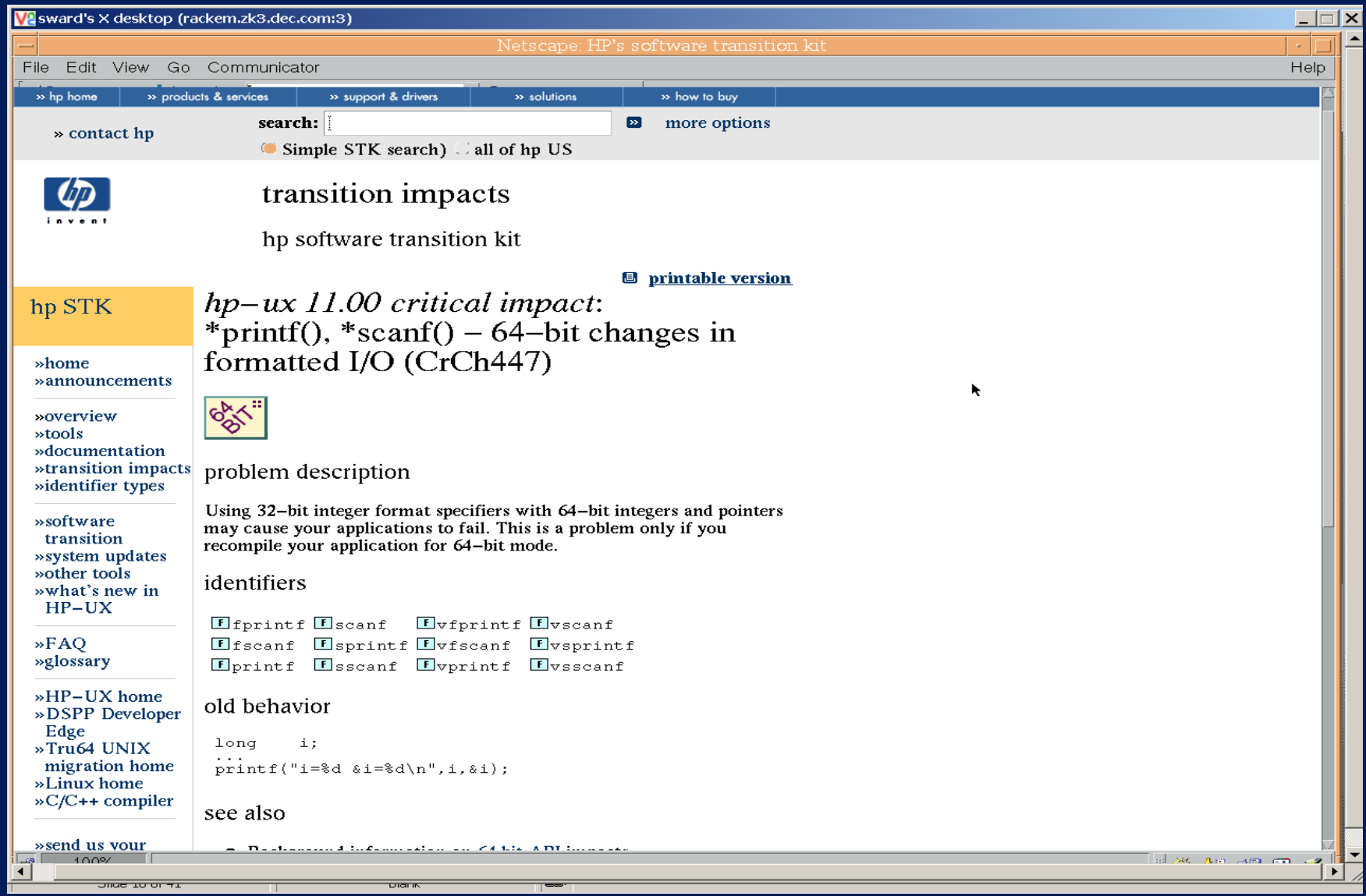
# Impact statement



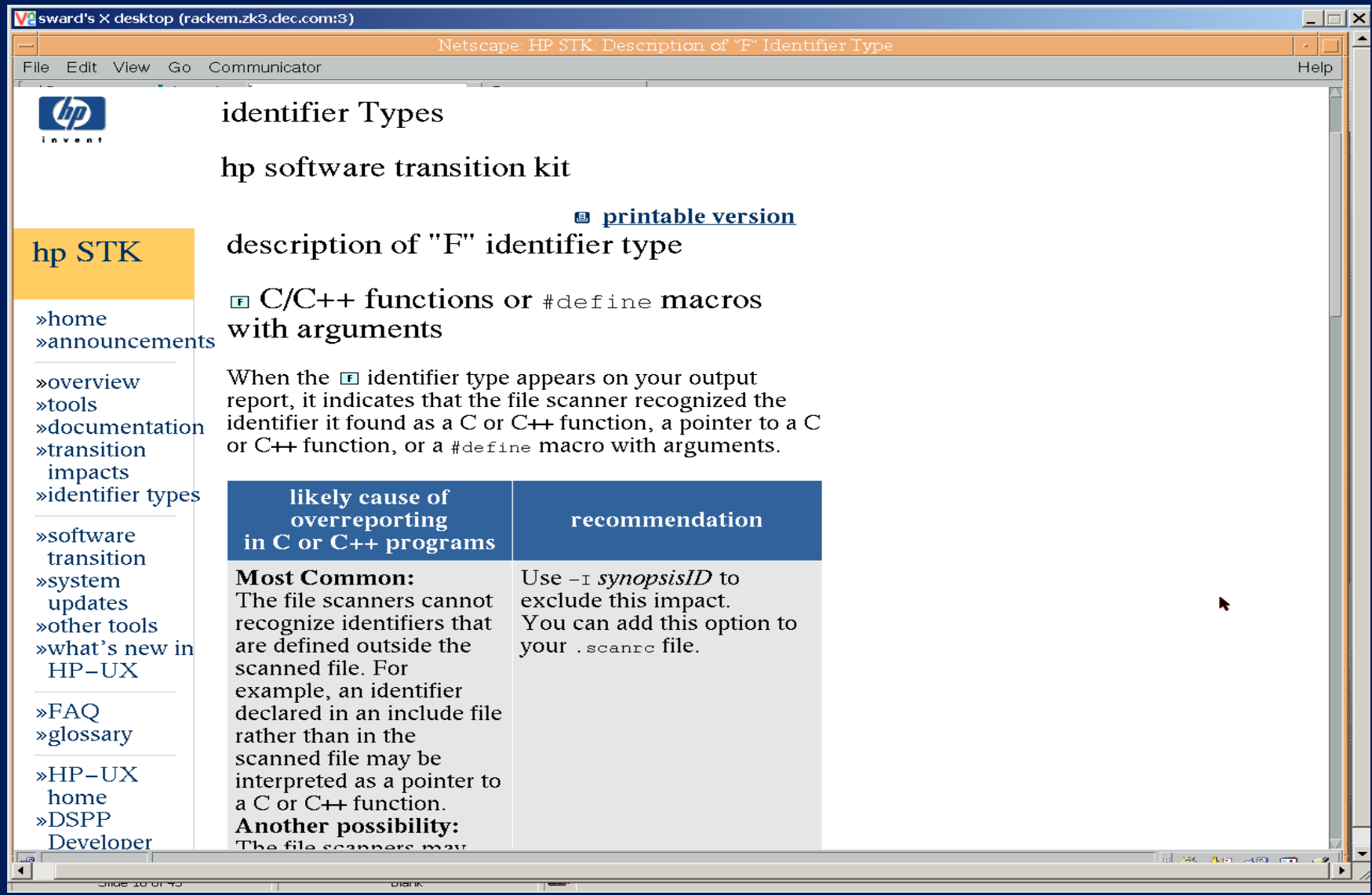
# Legend


sward's X desktop (rackem.zk3.dec.com:3)
Netscape: HP STK: 64–Bit API Impacts Summary
File Edit View Go Communicator Help
transition impacts
hp software transition kit
printable version
64–bit API impacts summary
hp STK
»home
»announcements
»overview
»tools
»documentation
»transition impacts
»identifier types
»software transition
»system updates
»other tools
»what's new in HP–UX
»FAQ
»glossary
»HP–UX home
»DSPP Developer
There are three types of code changes needed to create a 64–bit version of your software from your 32–bit clean source code:
convert from a 32–bit data model to a 64–bit data model
use the new 64–bit APIs
incorporate 64–bit features
32–bit and 64–bit UNIX computing use different data models. The data model used in 32–bit UNIX computing is called ILP32, meaning that integers, longs and pointers are all 32–bit data types. The data model used in 64–bit UNIX computing is called LP64, meaning that longs and pointers are 64–bit data types while integers remain as 32–bit data types. Some existing C and C++ code assumes that integers, longs and pointers are the same size. Because these assumptions are not true in LP64, some executables have defects when they are compiled for 64–bits.
Note
To include only 64–Bit API impacts when running the

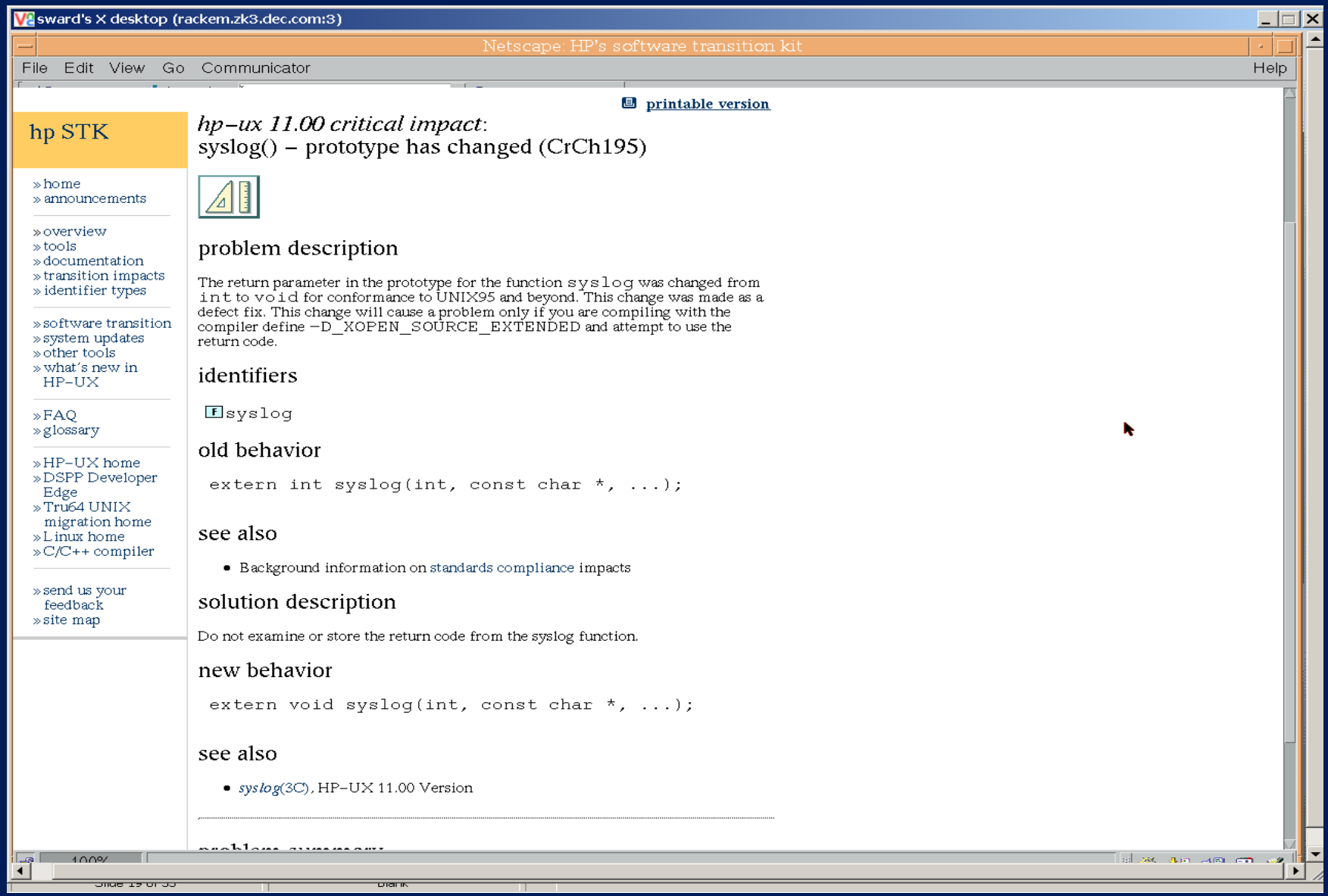
# Impact statement



# Identifier Type

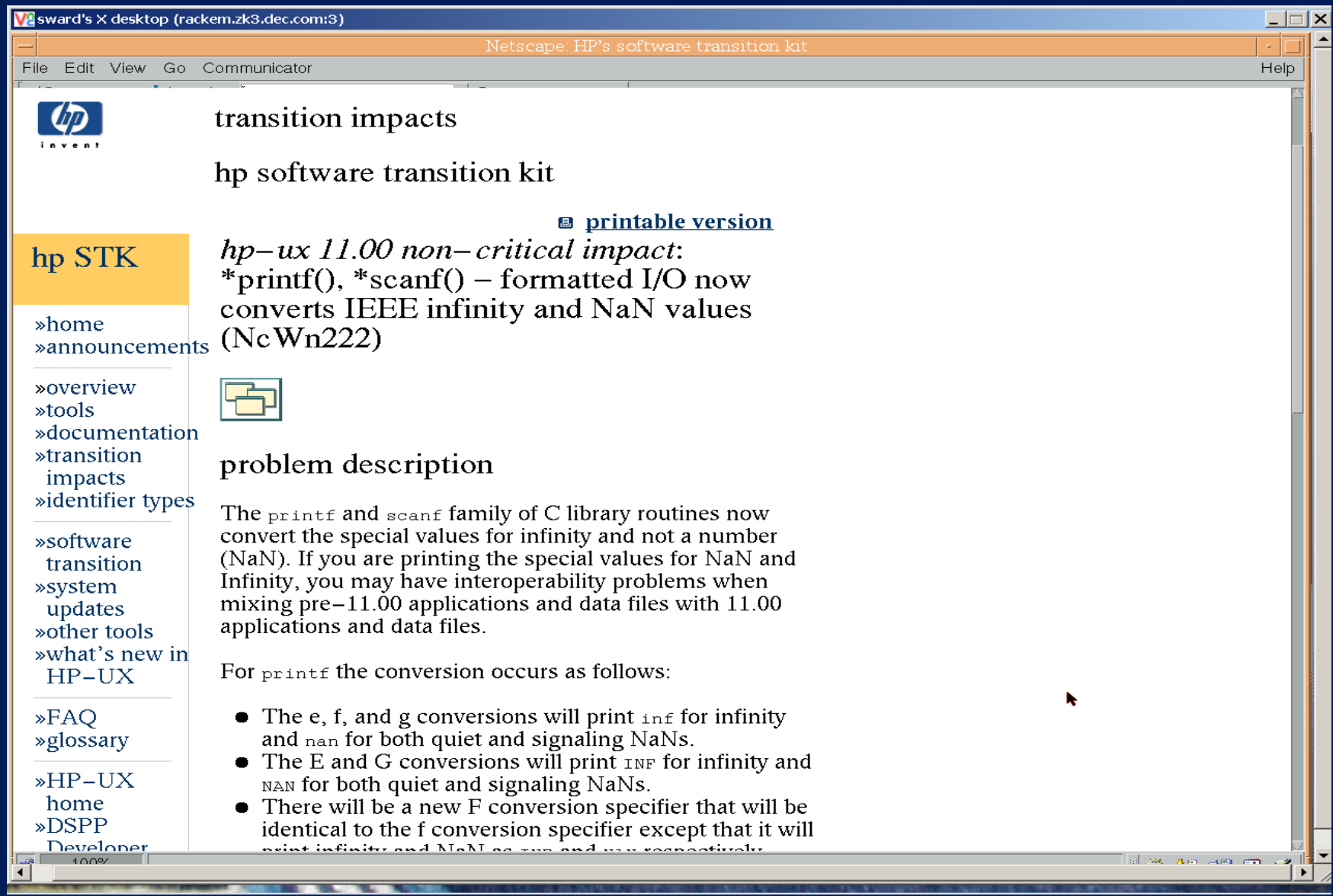


# Critical and Links



# Manpage link

# Non-Critical



# Code example (old)

sward's X desktop (rackem.zk3.dec.com:3)

Netscape: HP's software transition kit

File Edit View Go Communicator Help

transition impacts

hp software transition kit

printable version

hp STK

»home
»announcements

»overview
»tools
»documentation
»transition impacts
»identifier types

»software transition
»system updates
»other tools
»what's new in HP-UX

»FAQ
»glossary

»HP-UX home
»DSPP Developer Edge

*hp-ux 11.00 critical impact*:
*printf(), *scanf() – 64-bit changes in formatted I/O (CrCh447)


## problem description

Using 32-bit integer format specifiers with 64-bit integers and pointers may cause your applications to fail. This is a problem only if you recompile your application for 64-bit mode.

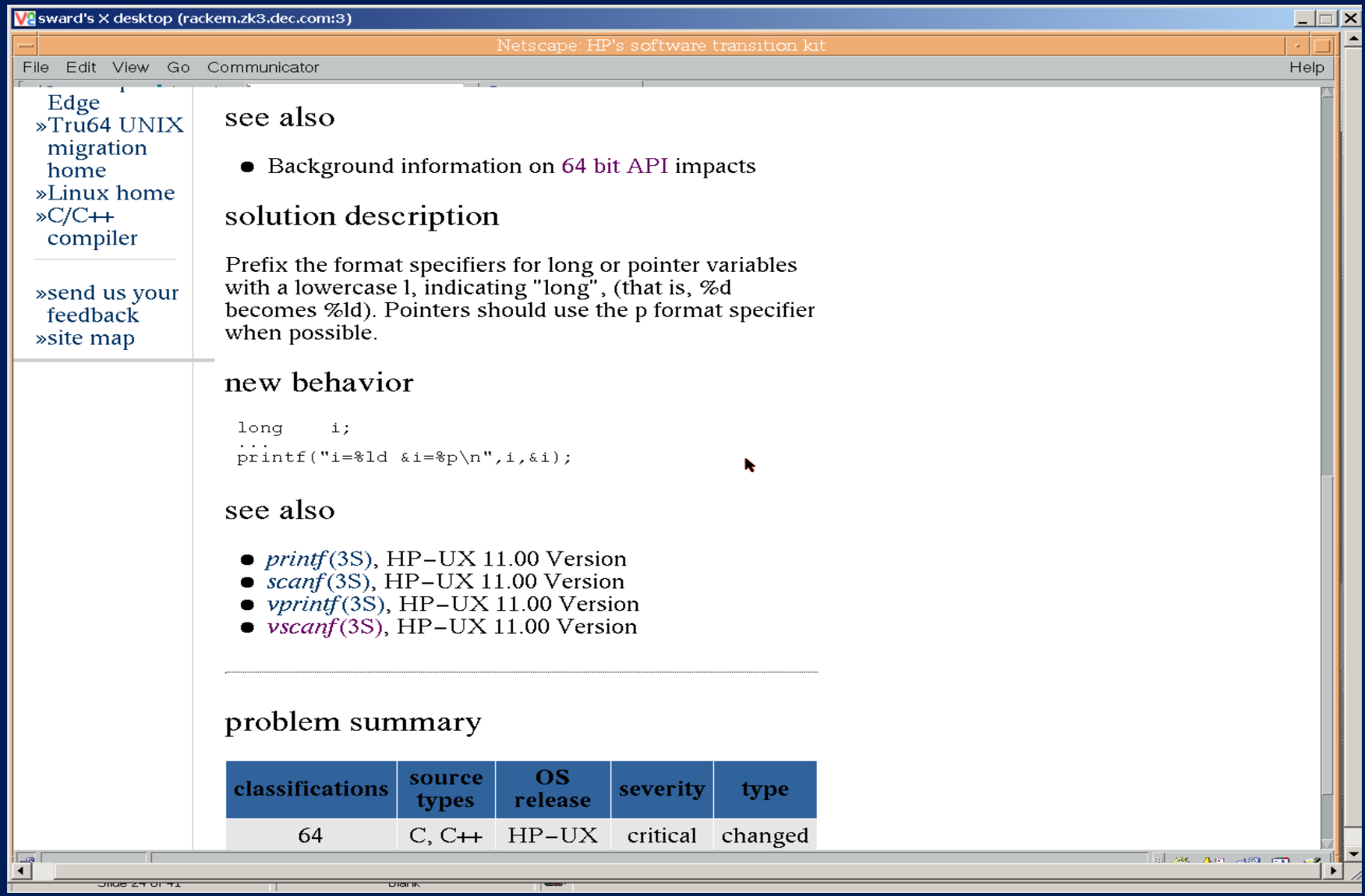
## identifiers

| | | | |
|---|---|---|---|
| fprintf | scanf | vfprintf | vscanf |
| fscanf | sprintf | vfscanf | vsprintf |
| printf | sscanf | vprintf | vsscanf |

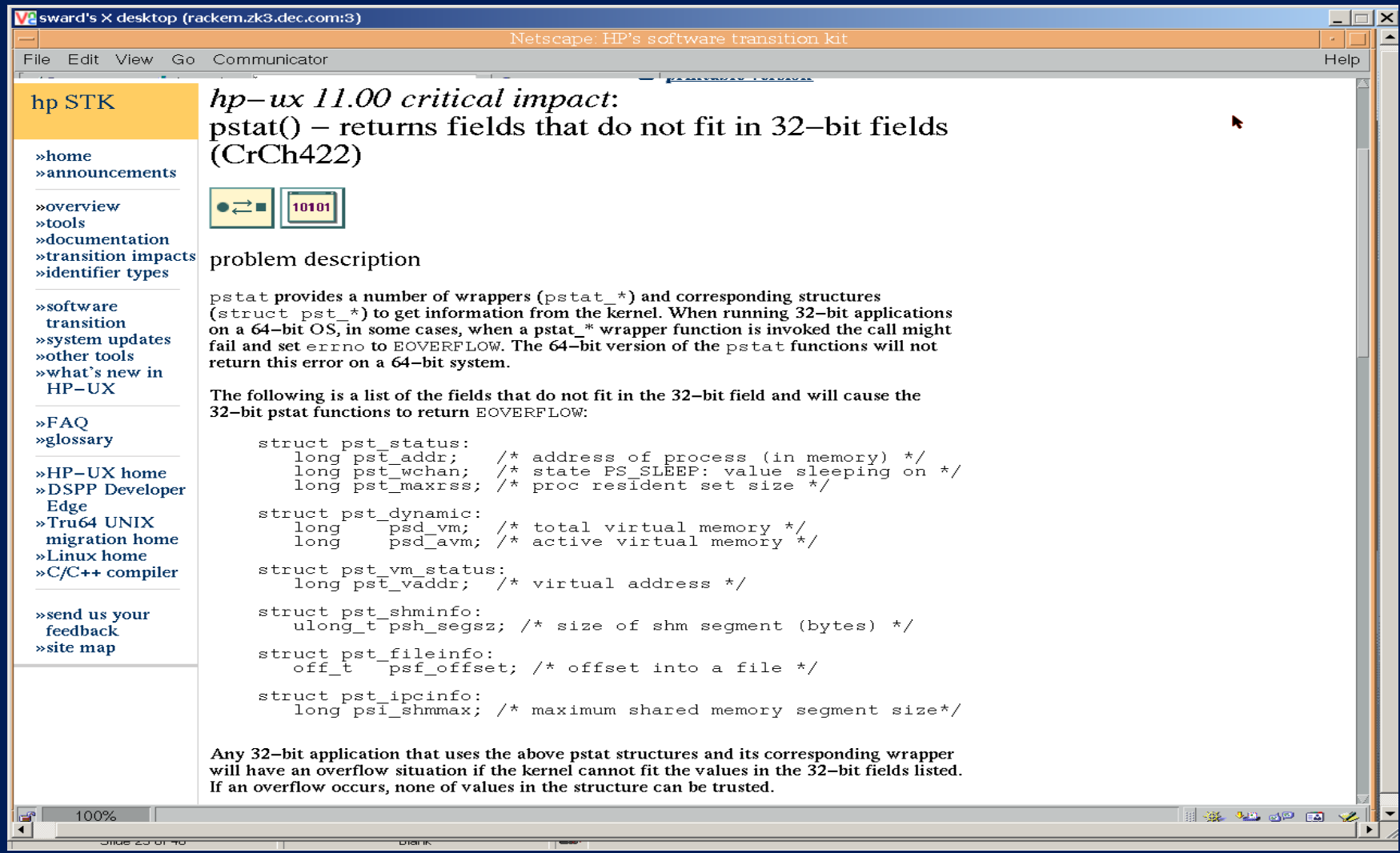
## old behavior

```
long    i;
...
printf("i=%d &i=%d\n",i,&i);
```

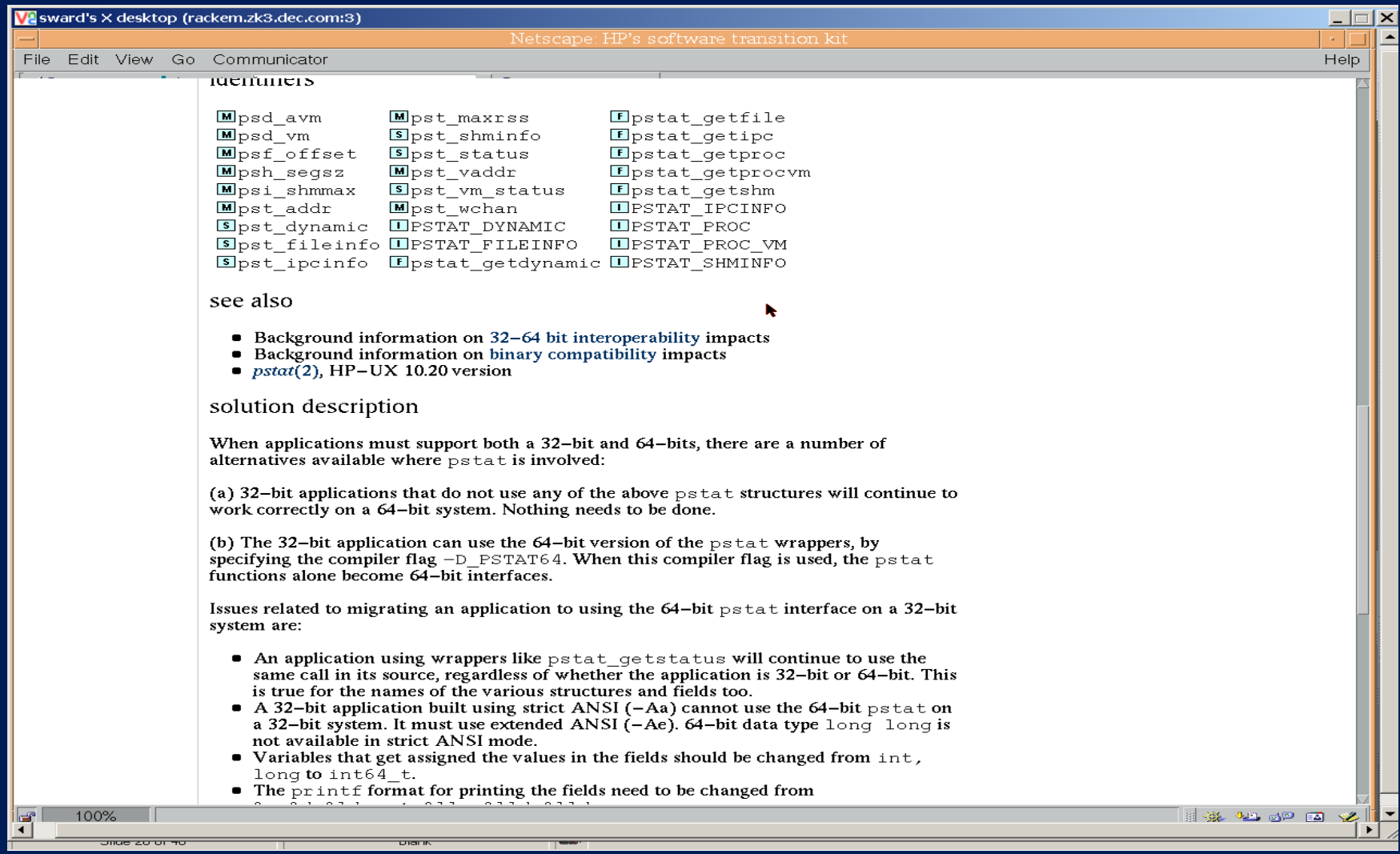
# Code example (new)



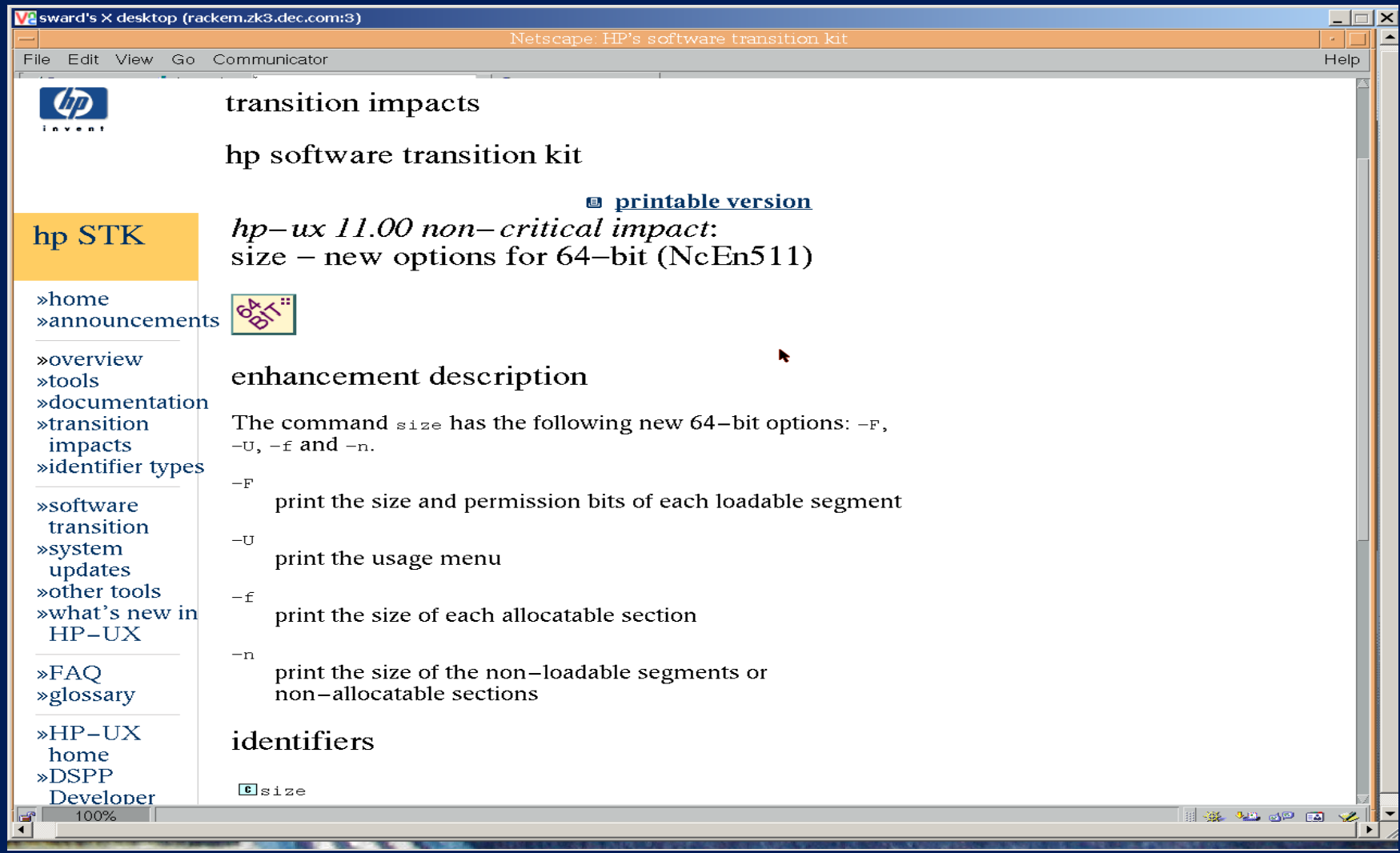
# Pstat (1/2)


sward's X desktop (rackem.zk3.dec.com:3)
Netscape: HP's software transition kit
File Edit View Go Communicator Help
hp STK
»home
»announcements
»overview
»tools
»documentation
»transition impacts
»identifier types
»software transition
»system updates
»other tools
»what's new in HP-UX
»FAQ
»glossary
»HP-UX home
»DSPP Developer Edge
»Tru64 UNIX migration home
»Linux home
»C/C++ compiler
»send us your feedback
»site map
hp-ux 11.00 critical impact:
pstat() – returns fields that do not fit in 32-bit fields (CrCh422)
10101
problem description
pstat provides a number of wrappers (pstat_*) and corresponding structures (struct pst_*) to get information from the kernel. When running 32-bit applications on a 64-bit OS, in some cases, when a pstat_* wrapper function is invoked the call might fail and set errno to EOVERFLOW. The 64-bit version of the pstat functions will not return this error on a 64-bit system.
The following is a list of the fields that do not fit in the 32-bit field and will cause the 32-bit pstat functions to return EOVERFLOW:
struct pst_status:
long pst_addr; /* address of process (in memory) */
long pst_wchan; /* state PS_SLEEP: value sleeping on */
long pst_maxrss; /* proc resident set size */
struct pst_dynamic:
long psd_vm; /* total virtual memory */
long psd_avm; /* active virtual memory */
struct pst_vm_status:
long pst_vaddr; /* virtual address */
struct pst_shminfo:
ulong_t psh_segsz; /* size of shm segment (bytes) */
struct pst_fileinfo:
off_t psf_offset; /* offset into a file */
struct pst_ipcinfo:
long psi_shmmax; /* maximum shared memory segment size*/
Any 32-bit application that uses the above pstat structures and its corresponding wrapper will have an overflow situation if the kernel cannot fit the values in the 32-bit fields listed. If an overflow occurs, none of values in the structure can be trusted.
100%

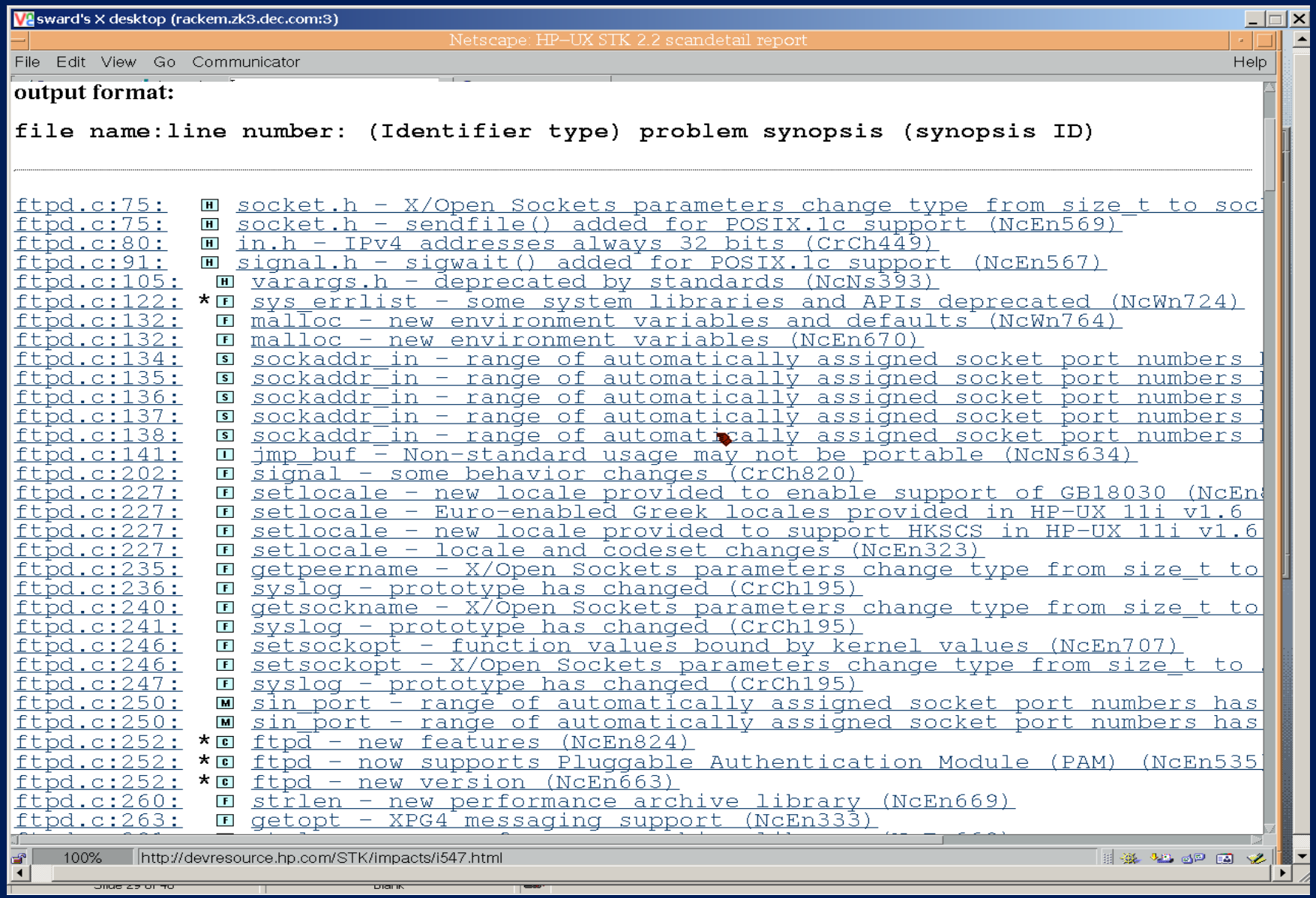
# pstat (2/2)


sward's X desktop (rackem.zk3.dec.com:3)
Netscape: HP's software transition kit
File Edit View Go Communicator Help
identifiers
M psd_avm
M psd_vm
M psf_offset
M psh_segsz
M psi_shmmax
M pst_addr
S pst_dynamic
S pst_fileinfo
S pst_ipcinfo
M pst_maxrss
S pst_shminfo
S pst_status
M pst_vaddr
S pst_vm_status
M pst_wchan
I PSTAT_DYNAMIC
I PSTAT_FILEINFO
F pstat_getdynamic
F pstat_getfile
F pstat_getipc
F pstat_getproc
F pstat_getprocvm
F pstat_getshm
I PSTAT_IPCINFO
I PSTAT_PROC
I PSTAT_PROC_VM
I PSTAT_SHMINFO
see also
Background information on 32-64 bit interoperability impacts
Background information on binary compatibility impacts
pstat(2), HP-UX 10.20 version
solution description
When applications must support both a 32-bit and 64-bits, there are a number of alternatives available where pstat is involved:
(a) 32-bit applications that do not use any of the above pstat structures will continue to work correctly on a 64-bit system. Nothing needs to be done.
(b) The 32-bit application can use the 64-bit version of the pstat wrappers, by specifying the compiler flag -D_PSTAT64. When this compiler flag is used, the pstat functions alone become 64-bit interfaces.
Issues related to migrating an application to using the 64-bit pstat interface on a 32-bit system are:
An application using wrappers like pstat_getstatus will continue to use the same call in its source, regardless of whether the application is 32-bit or 64-bit. This is true for the names of the various structures and fields too.
A 32-bit application built using strict ANSI (-Aa) cannot use the 64-bit pstat on a 32-bit system. It must use extended ANSI (-Ae). 64-bit data type long long is not available in strict ANSI mode.
Variables that get assigned the values in the fields should be changed from int, long to int64_t.
The printf format for printing the fields need to be changed from
100%

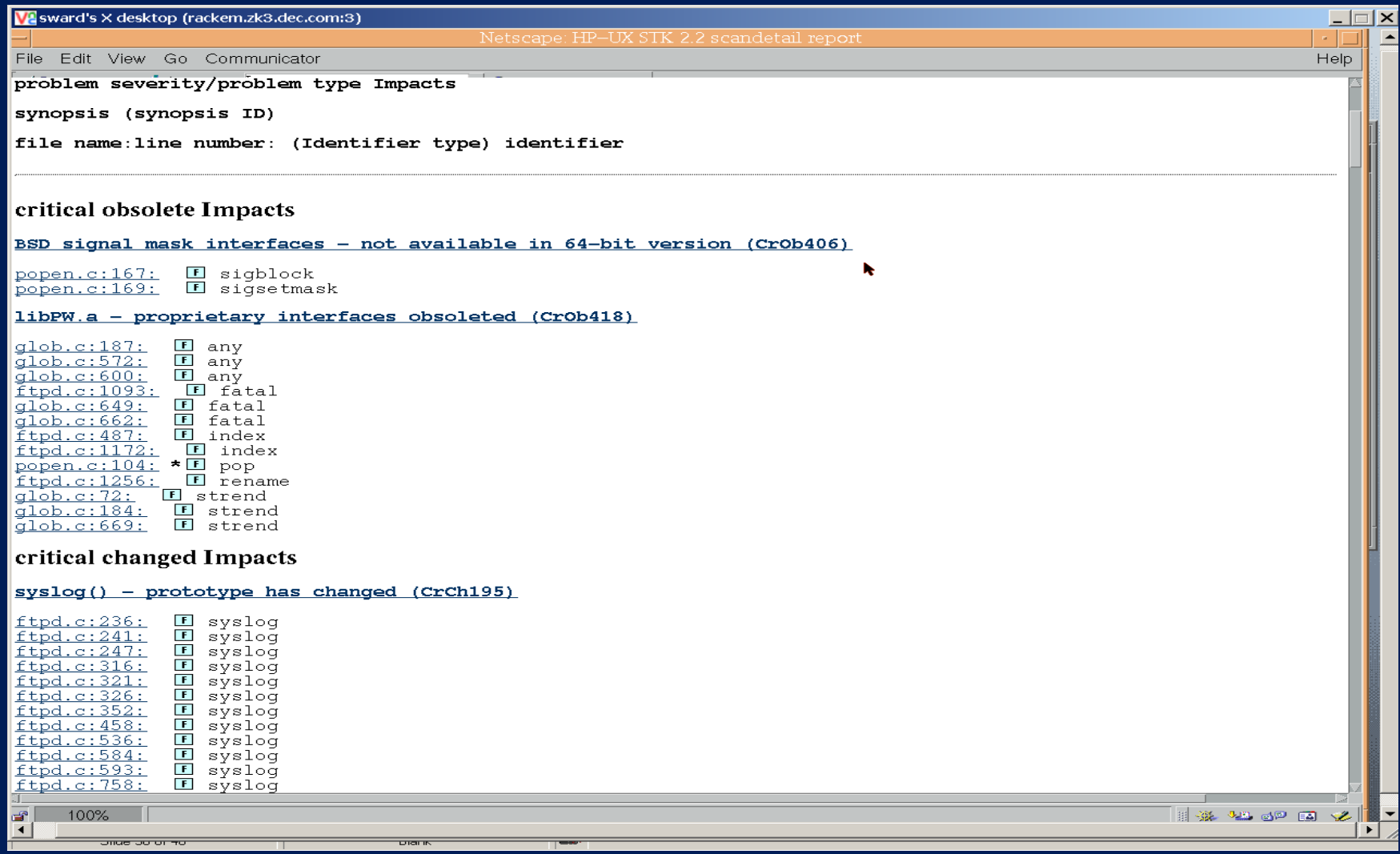
# Enhancements impact



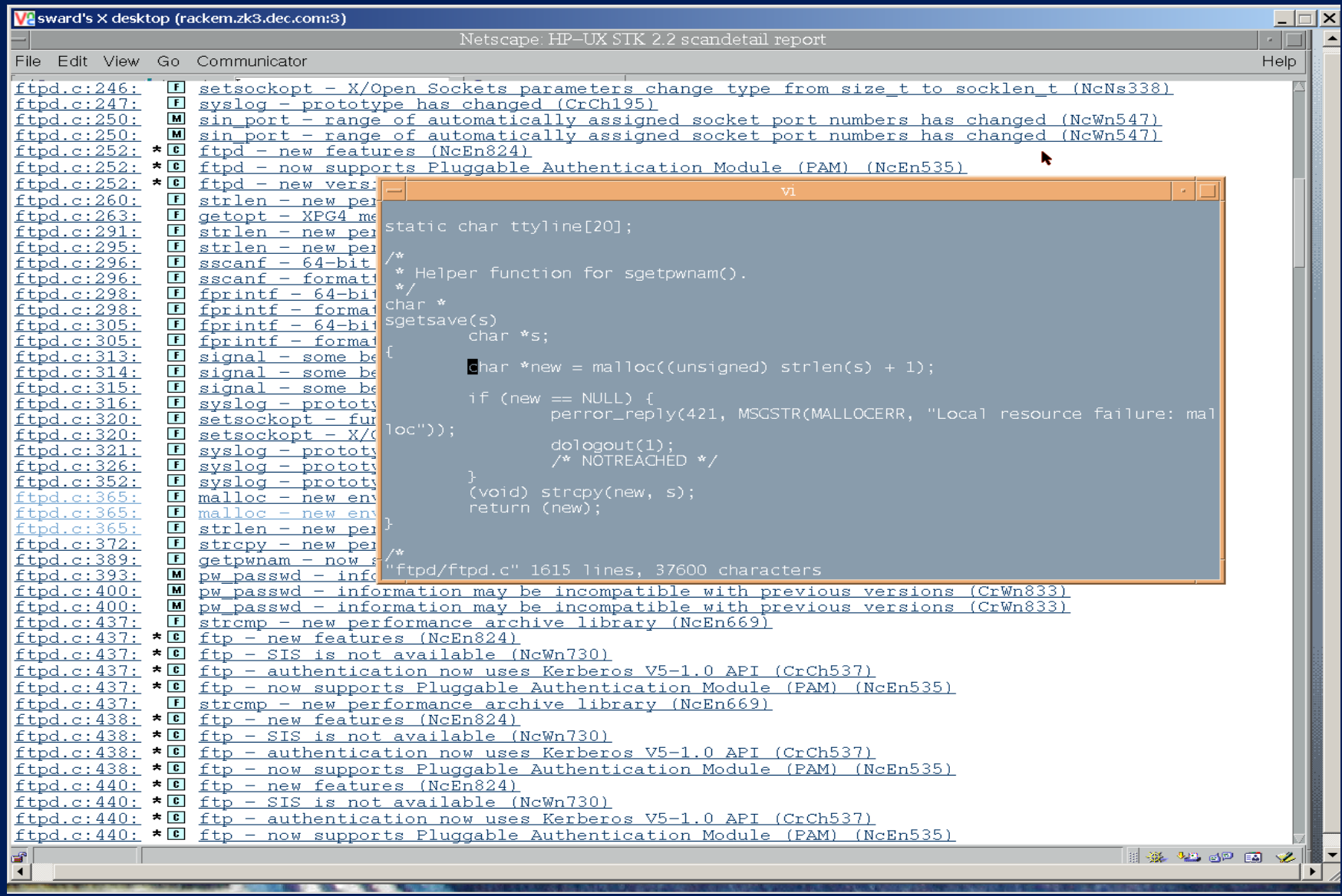
# scandetail


sward's X desktop (rackem.zk3.dec.com:3)
Netscape: HP-UX STK 2.2 scandetail report
File Edit View Go Communicator Help
output format:
file name:line number: (Identifier type) problem synopsis (synopsis ID)
ftpd.c:75: H socket.h - X/Open Sockets parameters change type from size_t to soc
ftpd.c:75: H socket.h - sendfile() added for POSIX.1c support (NcEn569)
ftpd.c:80: H in.h - IPv4 addresses always 32 bits (CrCh449)
ftpd.c:91: H signal.h - sigwait() added for POSIX.1c support (NcEn567)
ftpd.c:105: H varargs.h - deprecated by standards (NcNs393)
ftpd.c:122: * F sys_errlist - some system libraries and APIs deprecated (NcWn724)
ftpd.c:132: F malloc - new environment variables and defaults (NcWn764)
ftpd.c:132: F malloc - new environment variables (NcEn670)
ftpd.c:134: S sockaddr_in - range of automatically assigned socket port numbers
ftpd.c:135: S sockaddr_in - range of automatically assigned socket port numbers
ftpd.c:136: S sockaddr_in - range of automatically assigned socket port numbers
ftpd.c:137: S sockaddr_in - range of automatically assigned socket port numbers
ftpd.c:138: S sockaddr_in - range of automatically assigned socket port numbers
ftpd.c:141: I jmp_buf - Non-standard usage may not be portable (NcNs634)
ftpd.c:202: F signal - some behavior changes (CrCh820)
ftpd.c:227: F setlocale - new locale provided to enable support of GB18030 (NcEn
ftpd.c:227: F setlocale - Euro-enabled Greek locales provided in HP-UX 11i v1.6
ftpd.c:227: F setlocale - new locale provided to support HKSCS in HP-UX 11i v1.6
ftpd.c:227: F setlocale - locale and codeset changes (NcEn323)
ftpd.c:235: F getpeername - X/Open Sockets parameters change type from size_t to
ftpd.c:236: F syslog - prototype has changed (CrCh195)
ftpd.c:240: F getsockname - X/Open Sockets parameters change type from size_t to
ftpd.c:241: F syslog - prototype has changed (CrCh195)
ftpd.c:246: F setsockopt - function values bound by kernel values (NcEn707)
ftpd.c:246: F setsockopt - X/Open Sockets parameters change type from size_t to
ftpd.c:247: F syslog - prototype has changed (CrCh195)
ftpd.c:250: M sin_port - range of automatically assigned socket port numbers has
ftpd.c:250: M sin_port - range of automatically assigned socket port numbers has
ftpd.c:252: * C ftpd - new features (NcEn824)
ftpd.c:252: * C ftpd - now supports Pluggable Authentication Module (PAM) (NcEn535
ftpd.c:252: * C ftpd - new version (NcEn663)
ftpd.c:260: F strlen - new performance archive library (NcEn669)
ftpd.c:263: F getopt - XPG4 messaging support (NcEn333)
100%
http://devresource.hp.com/STK/impacts/i547.html

# Scandetail with –s synopsis



# Scandetail with editor



# Useful links

http://devresource.hp.com/STK For STK on HP-UX

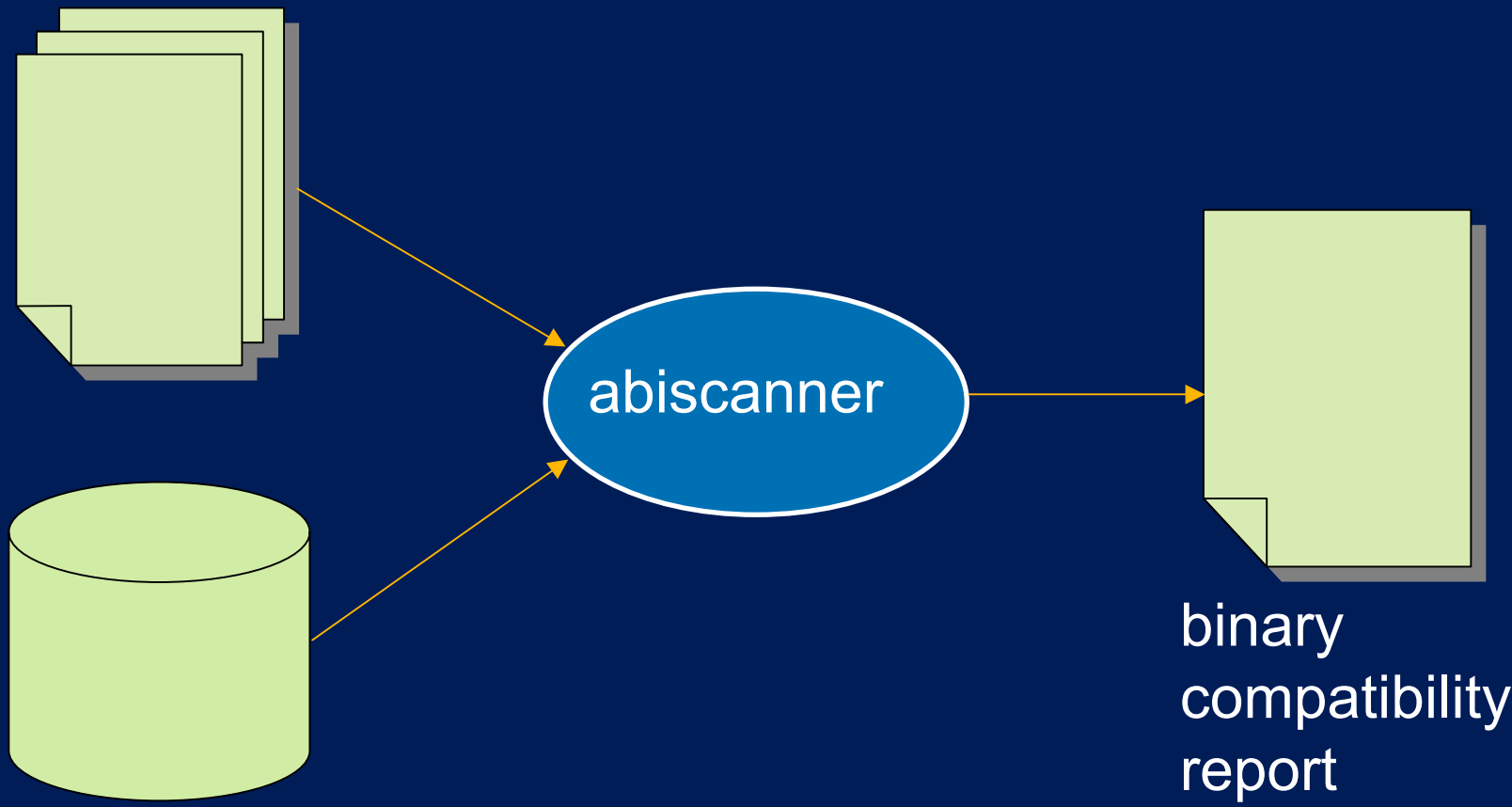
http://devresource.hp.com/STK_ja_JP.SJIS For Japanese

# abiscanner - overview

- A binary compatibility identification tool which reports whether a given target is compatible for a given platform
  - Scans all targets, which can be dynamically linked executables, libraries or object files
  - Reports the use of private Application Binary Interfaces (ABIs), for each target.
- Runs off a database defining the private/public ABIs(currently supports 11.0 to 11i.
- The scanner understands ELF and SOM formats
- Supported platforms : IA, PA
- Works on stripped binaries too

# abiscanner – how it works

Target files : executables, libraries, object files



# abiscanner – how it works

- Scan the target binaries/executables/libraries
- Get the list of libraries that were linked with the target executables and scan them
- Get the APIs imported by the target
- Foreach API, get the library providing that API (API/library mapping)
- Check the abi database to see if the API is listed and its classification(private/public /obsolete/..)
- Generate a report for all the APIs found

# abiscanner : Sample report

```
hpadl694:>abiscanner -o text -xl /usr/bin/login /usr/bin/ftp
/usr/bin/login:
Private    /usr/lib/libnsl.1    key_setnet
Private    /usr/lib/libpam.1    pam_get_username
/usr/bin/ftp:
Unidentified <null>             _LHcrPlVK
Unidentified <null>             _piY32JQN
/usr/bin/login (SOM Executable,PA-Risc 1.1,32-Bit):
Libraries:
[ 0] System         /usr/lib/libpam.1  PA-Risc 1.1, 32-Bit
[ 1] System         /usr/lib/libsec.2  PA-Risc 1.1, 32-Bit
[ 2] System         /usr/lib/libnsl.1  PA-Risc 1.1, 32-Bit
[ 3] System         /usr/lib/libc.2    PA-Risc 1.1, 32-Bit
/usr/bin/ftp (SOM Executable,PA-Risc 1.1,32-Bit):
Libraries:
[ 0] System         /usr/lib/libsis.sl PA-Risc 1.1, 32-Bit
[ 1] System         /usr/lib/libc.2    PA-Risc 1.1, 32-Bit
```

# abiscanner – classifications

ABIs are classified into different categories and are reported accordingly.

Current categories are :

- **Public** : supported and documented
- **Private** : undocumented and unsupported ABIs provided by HP-UX system libraries
- **Compatible private** : undocumented and unsupported but stable across versions
- **Obsolete** : will be unsupported in future

# abiscanner : Limitations

*abiscanner* **cannot** determine whether your application will work or not. It can only cross-reference interfaces in your application against the officially supported public/ private API list.

So …

- **If a component of your application has passed, then no uses of private interfaces were detected in that component. This does *not* mean that your application will work! It simply indicates that the component uses approved interfaces and passes the scanner criteria of the guarantee.**

- **If a component of your application has failed, then uses of private interfaces were detected in that component. This does *not* mean that your application will *not* work! It simply indicates that that component of the application uses private interfaces that are not supported by HP. The failing component should be tested.**

# abiscanner Links

Home :
http://devresource.hp.com/STK/

Quick Start:
http://devresource.hp.com/STK/hpux11i/quickstart.html

FAQs :
http://devresource.hp.com/STK/hpux11i/abiscannerFaq.html
FAQs :

Reference :
http://devresource.hp.com/STK/hpux11i/abiscannerRef.html
Reference :


invent