NAME

sysarch − defines system specific variables using the uname system command

SYNOPSIS

sysarch [OPTION]... [VARIABLE] [VARIABLE]

DESCRIPTION

sysarch provides a uniform description of a system, including the underlying hardware, the operating system, and the unique operating system characteristics. Its intent is to identify important system characteristics which determine the interoperability of software with hardware/operating system platforms.

Compatibility between operating systems sometimes occurs. Solaris is a true champion of this, retaining compatibility across more than a decade in some cases. It's also often possible to run programs compiled on one flavor of Linux on another. It is rare for software compiled on newer platforms to run on older ones, however. While **sysarch** can't determine which systems are compatible, it tries to present enough salient information so that it is possible to provide a hierarchy of compatibility.

sysarch encodes system information into a set of variables. It will generate code for various shells and GNU make which can be used to incorporate those variables into scripts and Makefiles.

OPTIONS AND ARGUMENTS

sysarch takes an optional argument specifying the name of a variable to output. If no options or arguments are specified they are all output.

General Options

−−export
Output code which creates environmental, rather than shell, variables. Only releveant for shells.

−−pfx *pfx*
Prepend *pfx* to variable names.

−−help
Print this usage information and exit.

Output Formats

Output formats are specified either using the **−−output** option, or simply by prefixing the format name with −− and treating it as an option:

```
--output csh
--csh
```

If no variables are specified on the command line, the default output format is `nice`. If variables are specified, it defaults to `value`.

The following formats are supported:

csh|**tcsh**|**sh**|**ksh**| **bash**
Output commands in the appropriate shell to create the variables.

make
Create static **make** compatible macro assignments. Macro names are in upper case.

gmake_extract
Create dynamic GNU **make** compatible macro assignments using a call to **sysarch**. Macro names are in upper case.

gmake_cat
Create the output required by the code output by `--output gmake_extract`.

make_dump
Generate **make** target commands to output the **sysarch** generated macros. Useful as part of a **make** `debug` style target.

perl
Output Perl code generating scalar variables.

perl_hash
Output Perl code for the contents of a hash. It does not generate the wrapper code required to actually create the hash.

value
Output only the variables' values.

nice
A more human readable format.

OUTPUT VARIABLES

Hardware Identification

hw_cpu
The most specific description of the actual CPU.

hw_cpu_generic
The generic description for the CPU. This is *not* the most generic one with which the CPU is compatible. For example,

```
athlon_64 => x86_64,  not i686
core2     => x86_64,  not i686
pentium4  => i686
ultra3i   => sun4u,   not sparc
```

hw_cpus
A colon separated list of models or families of CPU's with which the hardware is compatible, from least to most generic.

For example, for a SunBlade 1500,

```
hw_cpu  = ultra3i
hw_cpus = ultra3i:sun4u:sun4m:sparc
```

For an AMD Athlon64 CPU,

```
hw_cpu  = athlon_64
hw_cpus = athlon_64:x86_64:i686
```

System Identification

The system designation represents the combination of operating system, kernel, and optionally vendor or distribution name.

OS, **os_type**
The operating system name. **OS** has historical significance for Linux and Solaris, and retains the following value for them:

```
Linux
SunOS
```

For other operating systems, **OS** and **os_type** are equivalent. **os_type** is the lowercased more ''obvious'' name (i.e. `solaris` rather than `sunos`) and is typically equivalent to the **OSTYPE** environmental variable.

Note that `Linux` represents both the kernel and the accompanying system software. This is often represented as GNU/Linux, similar to Debian GNU/kFreeBSD, Debian GNU/NetBSD, GNU/Hurd and Debian GNU/Solaris. The latter set aren't available to the authors, so we're not worrying about them yet.

os_vendor
os_vendor is the vendor or distribution name. Some vendors provide multiple distributions (e.g. Red Hat Fedora Core, Red Hat Enterprise Linux) — **os_vendor** is then the combination (e.g. `RHFC`,

RHEL).

System Version

System version information is complex. It might refer to the version given to a particular distribution or release (such as Fedora Core 3 or 4). For some applications the kernel version is more important, or perhaps the version of glibc. **sysarch** provides all of that information.

kern_version

kern_version is the kernel version. On Linux it is the base kernel, without any vendor specific version information.

kern_version_maj

kern_version_maj contains just the most significant kernel version (e.g. 2.6 for Linux, 8.8 for Darwin).

kern_versions

A colon separated list of kernels with which this is compatible.

os_version, **os_version_maj**

os_version is the vendor supplied operation system version. **os_version_maj** is the most significant part (e.g, for OSX 10.4 vs. 10.4.8).

os_versions

A colon separated list of operating system versions with which this operating system is backwards compatible.

os_uversion, **os_uversion_maj**

The version of the operating system component most important for user space applications. For Linux this is chosen to be the version of glibc. This may not be restrictive enough for some applications (e.g. if one distribution ships with a different version of **libreadline**). **os_version_maj** is the most significant part (e.g., 2.3 vs. 2.3.6).

os_uversions

A colon separated list of user space component versions with which this operating system is backwards compatible. For Linux it is assumed that previous minor (in the glibc sense) releases are backwards compatible.

Platform

A platform is the combination of hardware and operating system specifications. It's a triad with one of the following combinations:

```
<hw_cpu>-<os_type>-<os_uversion>
<hw_cpu>-<OS>-<os_version>
```

where <OS> is

```
<os_type>_<os_vendor>
```

except for Solaris and OSX (Darwin) where it is

```
<os_type>
```

It is assumed that **os_uversion** is constant for a given **os_version**.

Host System

These variables describe the host system's platform

platform
platform_native

Aliases for **platform_os_native**.

platform_generic

An alias for **platform_uos_generic**.

platform_os_native
The most detailed specification including the OS vendor/distribution. Equivalent to

```
<hw_cpu>-<OS>-<os_version>
```

platform_os_generic
This specification allows for a more generic hardware and OS specification, allowing for compatibility within minor OS and hardware variations. This is equivalent to

```
<hw_cpu_generic>-<OS>-<os_version_maj>
```

platform_uos_native
The most detailed specification using the user space component version. Equivalent to

```
<hw_cpu>-<os_type>-<os_uversion>
```

platform_uos_generic
Equivalent to

```
<hw_cpu_generic>-<OS>-<os_uversion_maj>
```

Compatible platforms

These variables provide lists of compatible platforms

platforms
A colon separated list of compatible platforms, with hardware being the most important element for compatibility. This is a combination of **platforms_os** and **platforms_uos**.

platforms_os
A colon separated list of compatible platforms of the form

```
<hw_cpu>-<OS>-<os_version>
```

With Hardware being the most important element for compatibility.

platforms_uos
A colon separated list of compatible platforms of the form

```
<hw_cpu>-<os_type>-<os_uversion>
```

with hardware being the most important element for compatibility.

WHAT DOES THIS ALL MEAN?

Well, that depends upon how its being used. First off, use **platforms** and **sysarch**'s accompanying **syspathsubst** program to create compatibility hierarchies for executable and library search paths.

platform_generic is **not** the lowest common denominator. It's essentially the most generic specification that can be natively created (i.e. compiled) on the platform. Because it purports to specify user space compatibility, use it on Linux if you're sure that your application will run on multiple Linux distributions based solely on **glibc** compatibility. If, however, your application links against libraries in a combination that differs between distributions, you might as well use **platform_os_generic**.

COPYRIGHT AND LICENSE

AUTHORS

M. Tibbetts <mtibbetts@cfa.harvard.edu>

D. Jerius <djerius@cfa.harvard.edu>