Particular recipes

Missing d_type of struct dirent

Synopsis:

```
inputoutput.cpp:82:27: error: ‘struct dirent’ has no member named ‘d_type’; did you mean ‘d_name’?

                if (dirp->d_type == DT_REG)
                          ^~~~~~
inputoutput.cpp:82:37: error: ‘DT_REG’ was not declared in this scope
                if (dirp->d_type == DT_REG)
                                    ^~~~~~
```

Solution is to use dirfd and fstatat, as shown below.
Note that decent programs should use stat/fstat/fstatat anyway, because not checking d_type == DT_UNKNOWN *is a bug*!

```
--- opencv-2.4.9.1+dfsg1.orig/modules/contrib/src/inputoutput.cpp
+++ opencv-2.4.9.1+dfsg1/modules/contrib/src/inputoutput.cpp
@@ -7,6 +7,7 @@
     #include <tchar.h>
 #else
     #include <dirent.h>
+    #include <sys/stat.h>
 #endif

 namespace cv
@@ -72,14 +73,18 @@ namespace cv
             (void)addPath;
             DIR *dp;
             struct dirent *dirp;
+            int dfd;
+            struct stat st;
             if((dp = opendir(path.c_str())) == NULL)
             {
                 return list;
             }
+            dfd = dirfd(dp);

             while ((dirp = readdir(dp)) != NULL)
             {
-                if (dirp->d_type == DT_REG)
+                if ((0 == fstatat(dfd, dirp->d_name, &st, 0)) &&
+                    S_ISREG(st.st_mode))
                 {
                     if (exten.compare("*") == 0)
                         list.push_back(static_cast<std::string>(dirp->d_name));
```

struct ifreq

If struct ifreq causes problems (e. g. missing ifr_ifindex) consider using struct lifreq, see for example http://bugs.python.org/issue21287.

Too many arguments to function ‘getpwuid_r’, etc.

Compile with CPPFLAGS="-D_POSIX_C_SOURCE=199506L" or "-D_POSIX_PTHREAD_SEMANTICS -D_REENTRANT"

Missing strverscmp()

Use libiberty-dev package: link with -liberty, include libiberty/libiberty.h. The only known package with such a problem is tree (actually it includes strverscmp.c).

Debian symbols file

glibc defines FILE as _IO_FILE, illumos libc - as __FILE.
So C++ functions differ, and dh_makeshlibs can fail:

```
dpkg-gensymbols: warning: some new symbols appeared in the symbols file: see diff output below
dpkg-gensymbols: warning: some symbols or patterns disappeared in the symbols file: see diff outpu
t below
dpkg-gensymbols: warning: debian/libqt5webkit5/DEBIAN/symbols doesn't match completely debian/libq
t5webkit5.symbols
...
dh_makeshlibs: failing due to earlier errors
```

Fix:

```
@@ -2182,7 +2182,8 @@ libQt5WebKit.so.5 libqt5webkit5 #MINVER#
  _ZN3JSC7JSProxy9setTargetERNS_2VMEPNS_14JSGlobalObjectE@Base 5.2.0
  _ZN3JSC7JSScope13objectAtScopeEPS0_@Base 5.0.2
  _ZN3JSC7JSValue13isValidCalleeEv@Base 5.0.2
- _ZN3JSC7Options14dumpAllOptionsEP8_IO_FILE@Base 5.0.2
+ (arch=!illumos-amd64)_ZN3JSC7Options14dumpAllOptionsEP8_IO_FILE@Base 5.0.2
+ (arch=illumos-amd64)_ZN3JSC7Options14dumpAllOptionsEP6__FILE@Base 5.2.1+dfsg-6~dyson1
  _ZN3JSC7Options9s_optionsE@Base 5.0.2
  _ZN3JSC7Options9setOptionEPKc@Base 5.0.2
  _ZN3JSC7Profile10restoreAllEv@Base 5.0.2
```

Getting stack boundaries

On Linux we usually use pthread_getattr_np(), on Dyson we have thr_stksegment().

Be careful: if stack is "unlimited", ss_size will hold RLIM_INFINITY = -3 (see sys/resource.h) which definitely must not be interpreted as stack size! In this case lower stack boundary should be set to 0xFFFF800000000000L.

Example from Mono:

```
#elif (defined(__sun__) || defined(__illumos_kernel__))
#include <thread.h>
#include <errno.h>
    {
        int rc;
        stack_t s;

        do {
            rc = thr_stksegment(&s);
        } while (rc != 0 && errno == EAGAIN);
```

```
        info->stack_end = s.ss_sp;

        // Is stack "unlimited" ?
        if (s.ss_size == RLIM_INFINITY)
        {
            // XXX Lower stack boundary
            info->stack_start_limit = (void*)0xFFFF800000000000L;
        }
        else
        {
            info->stack_start_limit = (void*)((intptr_t)s.ss_sp - (intptr_t)s.ss_size);
        }
    }
#else
```

P. S. pthread_attr_get_np() is provided by libc since 5.10.18.git.2a44663-4 (2017-11-22).

Getting path of executable

Use const char * getexecname() from stdlib.h or

```
readlink("/proc/self/path/a.out", buf, bufsize);
```

http://stackoverflow.com/questions/933850/how-to-find-the-location-of-the-executable-in-c

Conflicting declarations of abs() in C++ sources

Synopsis:

```
In file included from ../../src/base/dict.h:40:0,
                 from freetype.cpp:152:
/usr/include/c++/4.8/cstdlib: In function ‘long long int std::abs(long long int)’:
/usr/include/c++/4.8/cstdlib:174:20: error: declaration of C function ‘long long int std::abs(long
 long int)’ conflicts with
   abs(long long __x) { return __builtin_llabs (__x); }
                    ^
In file included from /usr/include/stdlib.h:38:0,
                 from ../../include/Inventor/SoType.h:37,
                 from ../../include/Inventor/errors/SoError.h:39,
                 from ../../include/Inventor/errors/SoDebugError.h:36,
                 from ../../src/coindefs.h:95,
                 from freetype.cpp:44:
/usr/include/iso/stdlib_iso.h:122:12: error: previous declaration ‘int std::abs(int)’ here
 extern int abs(int);
            ^
```

Make sure C++ sources include cstdlib, not stdlib.h.

Missing IXDR_GET_LONG or similar *_LONG RPC macros

These macros are deprecated and removed from 64-bit Solaris or illumos. Glibc provides them via *_INT32 macros. Workaround:

```
#include <rpc/xdr.h>

// ...

#ifndef IXDR_GET_LONG
#define IXDR_GET_LONG(buf) ((long)IXDR_GET_U_INT32(buf))
```

```
#endif

#ifndef IXDR_PUT_LONG
#define IXDR_PUT_LONG(buf, v) ((long)IXDR_PUT_INT32(buf, (long)(v)))
#endif

#ifndef IXDR_GET_U_LONG
#define IXDR_GET_U_LONG(buf) ((u_long)IXDR_GET_LONG(buf))
#endif

#ifndef IXDR_PUT_U_LONG
#define IXDR_PUT_U_LONG(buf, v) IXDR_PUT_LONG(buf, (long)(v))
#endif
```

Boost threads unavailable on this platform

See /usr/include/boost/config/platform/solaris.hpp: -D_PTHREADS must be defined, use g++ -pthread