Unknown Applicability (C Rules/Recomendations)

The first two tables list CERT C Coding Standard rules and recommendations that are *Applicable in Principle,* meaning that the guideline can be applied to Android app development, but examples currently shown in the guideline are not yet relevant to Android. The third table lists rules and recommendations with *Unknown* applicability, meaning they are guidelines whose applicability to Android app development has not yet been determined.

Contents

Rules/Unknown Applicability to Android Development

Rules	Comments
EXP35-C. Do not modify objects with temporary lifetime	Possibly C11 relevant material, relevance must be considered on Android.
INT36-C. Converting a pointer to integer or integer to pointer	Don't know if a hardware platform for Android that this applies to. (only apply to certain arcane platform)
STR34-C. Cast characters to unsigned char before converting to larger integer sizes	Can chars be unsigned on Android? Might be a compiler option, so yes. Not needed for new code, but might have previous code affected by it.
STR38-C. Do not confuse narrow and wide character strings and functions	Not sure, needs more investigation.
FIO29-C. Do not open a file that is already open	
FIO32-C. Do not perform operations on devices that are only appropriate for files	
FIO34-C. Distinguish between characters read from a file and EOF or WEOF	EOF/WEOF: Only apply to app's public files? Others protected by VM?
FIO37-C. Do not assume that fgets() or fgetws() returns a nonempty string when successful	
FIO38-C. Do not copy a FILE object	FIO reference. What is Android filesystem? http://stackoverflow.com/questions/2421826/what-is-androids-file-system It depends on what filesystem, for example /system and /data are yaffs2 while /sdcard is vfat By default, it uses YAFFS - Yet Another Flash File System. Depends on what hardware/platform you use. Since Android uses the Linux-kernel at this level, it is more or less possible to use whatever filesystem the Linux-kernel supports. But since most phones use some kind of nand flash, it is safe to assume that they use YAFFS. But please note that if some vendor wants to sell a Android netbook (with a harddrive), they could use ext3 or something like that.
FIO40-C. Reset strings on fgets() or fgetws() failure	
FIO41-C. Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects	

FIO42-C. Close files when they are no longer needed	
FIO44-C. Only use values for fsetpos() that are returned from fgetpos()	
FIO45-C. Avoid TOCTOU race conditions while accessing files	
FIO46-C. Do not access a closed file	
CON30-C. Clean up thread-specific storage	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON31-C. Do not destroy a mutex while it is locked	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON32-C. Prevent data races when accessing bit-fields from multiple threads	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON33-C. Avoid race conditions when using library functions	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON34-C. Declare objects shared between threads with appropriate storage durations	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON35-C. Avoid deadlock by locking in a predefined order	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON36-C. Wrap functions that can spuriously wake up in a loop	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON37-C. Do not call signal() in a multithreaded program	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON38-C. Preserve thread safety and liveness when using condition variables	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON39-C. Do not join or detach a thread that was previously joined or detached	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON40-C. Do not refer to an atomic variable twice in an expression	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON41-C. Wrap functions that can fail spuriously in a loop	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
POS30-C. Use the readlink() function properly	

POS33-C. Do not use vfork()	
POS34-C. Do not call putenv() with a pointer to an automatic variable as the argument	
POS35-C. Avoid race conditions while checking for the existence of a symbolic link	Can apps get root? Can they run a shell? These rules/guidelines are not meant to address rooted devices (different OS than standard Android)
POS36-C. Observe correct revocation order while relinquishing privileges	Can apps get root? Can they run a shell? These rules/guidelines are not meant to address rooted devices (different OS than standard Android)
POS37-C. Ensure that privilege relinquishment is successful	Can apps get root? Can they run a shell? These rules/guidelines are not meant to address rooted devices (different OS than standard Android)
POS38-C. Beware of race conditions when using fork and file descriptors	
POS39-C. Use the correct byte ordering when transferring data between systems	
POS44-C. Do not use signals to terminate threads	Further investigation needed, specific to SigAction
POS47-C. Do not use threads that can be canceled asynchronously	
POS48-C. Do not unlock or destroy another POSIX thread's mutex	
POS49-C. When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed	Look into Android specifics for this, regarding "guarantee no adjacent data is accessed".
POS50-C. Declare objects shared between POSIX threads with appropriate storage durations	
POS51-C. Avoid deadlock with POSIX threads by locking in predefined order	
POS52-C. Do not perform operations that can block while holding a POSIX lock	

POS53-C. Do not use more than one mutex for concurrent waiting operations on a condition variable	
POS54-C. Detect and handle POSIX library errors	

Recommendations/Unknown Applicability to Android Development

Recommendations	Comments
ARR00-C. Understand how arrays work	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR01-C. Do not apply the sizeof operator to a pointer when taking the size of an array	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR02-C. Explicitly specify array bounds, even if implicitly defined by an initializer	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR30-C. Do not form or use out-of-bounds pointers or array subscripts	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR32-C. Ensure size arguments for variable length arrays are in a valid range	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR36-C. Do not subtract or compare two pointers that do not refer to the same array	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR37-C. Do not add or subtract an integer to a pointer to a non-array object	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR38-C. Guarantee that library functions do not form invalid pointers	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
ARR39-C. Do not add or subtract a scaled integer to a pointer	Arrays: Need examination of Android support. (gcc support of arrays partial). Note native code array issues different with ART than with Dalvik: http://developer.android.com/guide/practices/verifying-apps-art.html#JNI_Issues
FIO01-C. Be careful using functions that use file names for identification	
FIO02-C. Canonicalize path names originating from tainted sources	
FIO03-C. Do not make assumptions about fopen() and file creation	
FIO05-C. Identify files using multiple file attributes	
FIO06-C. Create files with appropriate access permissions	
FIO08-C. Take care when calling remove() on an open file	
FIO09-C. Be careful with binary data when transferring data across systems	
FIO10-C. Take care when using the rename() function	

FIO11-C. Take care when specifying the mode parameter of fopen()	
FIO13-C. Never push back anything other than one read character	
FIO14-C. Understand the difference between text mode and binary mode with file streams	
FIO15-C. Ensure that file operations are performed in a secure directory	FIO15-C: Look at Android/C interactions more closely to see if applicable.
FIO17-C. Do not rely on an ending null character when using fread()	
FIO18-C. Never expect fwrite() to terminate the writing process at a null character	
FIO19-C. Do not use fseek() and ftell() to compute the size of a regular file	
FIO20-C. Avoid unintentional truncation when using fgets() or fgetws()	
FIO21-C. Do not create temporary files in shared directories	
FIO22-C. Close files before spawning processes	
FIO23-CFIO23-C. Do not exit with unflushed data in stdout or stderr	
API01-C. Avoid laying out strings in memory directly before sensitive data	
API09-C. Compatible values should have the same type	
CON43-C. Do not allow data races in multithreaded code	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON01-C. Acquire and release synchronization primitives in the same module, at the same level of abstraction	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON02-C. Do not use volatile as a synchronization primitive	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON03-C. Ensure visibility when accessing shared variables	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON04-C. Join or detach threads even if their exit status is unimportant	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON05-C. Do not perform operations that can block while holding a lock	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON06-C. Ensure that every mutex outlives the data it protects	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON07-C. Ensure that compound operations on shared variables are atomic	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON08-C. Do not assume that a group of calls to independently atomic methods is atomic	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
CON09-C. Avoid the ABA problem when using lock-free algorithms	CON concurrency: Need to look into Android, does it support C11? Look at specifics of Android. Maybe POSIX threads, not C11 threads.
POS01-C. Check for the existence of links when dealing with files	
POS04-C. Avoid using PTHREAD_MUTEX_NORMAL type mutex locks	
POS05-C. Limit access to files by creating a jail	