Shellcheck Error Codes

SC1000	$ is not used specially and should therefore be escaped.
SC1001	This \c will be a regular 'c' in this context.
SC1007	Remove space after = if trying to assign a value (or for empty string, use var='' ...).
SC1009	The mentioned parser error was in ...
SC1010	Use semicolon or linefeed before 'done' (or quote to make it literal).
SC1015	This is a unicode double quote. Delete and retype it.
SC1016	This is a unicode single quote. Delete and retype it.
SC1018	This is a unicode non-breaking space. Delete it and retype as space.
SC1035	You need a space here
SC1037	Braces are required for positionals over 9, e.g. ${10}.
SC1038	Shells are space sensitive. Use '< <(cmd)', not '<<(cmd)'.
SC1040	When using <<-, you can only indent with tabs.
SC1045	It's not 'foo &; bar', just 'foo & bar'.
SC1065	Trying to declare parameters? Don't. Use () and refer to params as $1, $2..
SC1066	Don't use $ on the left side of assignments.
SC1068	Don't put spaces around the = in assignments.
SC1072	Unexpected ..
SC1073	Couldn't parse this ...
SC1077	For command expansion, the tick should slant left (` vs ´).
SC1078	Did you forget to close this double quoted string?
SC1079	This is actually an end quote, but due to next char it looks suspect.
SC1081	Scripts are case sensitive. Use 'if', not 'If'.
SC1083	This {/} is literal. Check expression (missing ;/\n?) or quote it.
SC1084	Use #!, not !#, for the shebang.
SC1086	Don't use $ on the iterator name in for loops.
SC1087	Braces are required when expanding arrays, as in ${array[idx]}.
SC1088	Parsing stopped here. Invalid use of parentheses?
SC1089	Parsing stopped here. Is this keyword correctly matched up?
SC1090	Can't follow non-constant source. Use a directive to specify location.
SC1091	Not following: (error message here)
SC1094	Parsing of sourced file failed. Ignoring it.
SC1095	You need a space or linefeed between the function name and body.
SC1097	Unexpected ==. For assignment, use =. For comparison, use [/[[.
SC1098	Quote/escape special characters when using eval, e.g. eval "a=(b)".
SC1099	You need a space before the #.
SC2001	SC2001: See if you can use ${variable//search/replace} instead.
SC2002	Useless cat. Consider 'cmd < file | ..' or 'cmd file | ..' instead.
SC2003	expr is antiquated. Consider rewriting this using $((..)), ${} or .
SC2004	$/${} is unnecessary on arithmetic variables.
SC2005	Useless echo? Instead of echo $(cmd), just use cmd
SC2006	Use $(..) instead of legacy \`..\`
SC2008	echo doesn't read from stdin, are you sure you should be piping to it?
SC2009	SC2009 Consider using pgrep instead of grepping ps output.
SC2010	Don't use ls | grep. Use a glob or a for loop with a condition to allow non-alphanumeric filenames.
SC2012	Use find instead of ls to better handle non-alphanumeric filenames.
SC2013	To read lines rather than words, pipe/redirect to a 'while read' loop.
SC2014	This will expand once before find runs, not per file found.
SC2015	Note that A && B || C is not if-then-else. C may run when A is true.
SC2016	Expressions don't expand in single quotes, use double quotes for that.
SC2017	Increase precision by replacing a/b*c with a*c/b.
SC2020	tr replaces sets of chars, not words (mentioned due to duplicates).
SC2021	Don't use [] around ranges in tr, it replaces literal square brackets.
SC2022	Note that unlike globs, o* here matches 'ooo' but not 'oscar'
SC2024	sudo doesn't affect redirects. Use ..| sudo tee file
SC2025	Make sure all escape sequences are enclosed in [..] to prevent line wrapping issues
SC2026	This word is outside of quotes. Did you intend to 'nest '"'single quotes'"' instead'?
SC2027	The surrounding quotes actually unquote this. Remove or escape them.
SC2028	echo won't expand escape sequences. Consider printf.
SC2029	Note that, unescaped, this expands on the client side.
SC2030	Modification of var is local (to subshell caused by pipeline).
SC2031	var was modified in a subshell. That change might be lost.
SC2032	Use own script or sh -c '..' to run this from su.
SC2033	Shell functions can't be passed to external commands.
SC2034	foo appears unused. Verify it or export it.
SC2035	Use ./*glob* or -- *glob* so names with dashes won't become options.
SC2036	If you wanted to assign the output of the pipeline, use a=$(b | c) .
SC2038	Use -print0/-0 or find -exec + to allow for non-alphanumeric filenames.
SC2039	In POSIX sh, *something* is undefined.
SC2040	#!/bin/sh was specified, so ____ is not supported, even when sh is actually bash.
SC2041	This is a literal string. To run as a command, use $(seq 1 10)
SC2043	This loop will only run once, with var=value
SC2044	For loops over find output are fragile. Use find -exec or a while read loop.
SC2045	Iterating over ls output is fragile. Use globs.
SC2046	Quote this to prevent word splitting
SC2048	Use "$@" (with quotes) to prevent whitespace problems.
SC2051	Bash doesn't support variables in brace range expansions.
SC2055	You probably wanted && here
SC2059	Don't use variables in the printf format string. Use printf "..%s.." "$foo".
SC2060	Quote parameters to tr to prevent glob expansion.
SC2061	Quote the parameter to -name so the shell won't interpret it.
SC2062	Quote the grep pattern so the shell won't interpret it.
SC2063	Grep uses regex, but this looks like a glob.
SC2064	Use single quotes, otherwise this expands now rather than when signaled.
SC2065	This is interpreted as a shell file redirection, not a comparison.
SC2066	Since you double quoted this, it will not word split, and the loop will only run once.
SC2067	Missing ';' or + terminating -exec. You can't use |/||/&&, and ';' has to be a separate, quoted argument.
SC2068	Double quote array expansions to avoid re-splitting elements.
SC2069	The order of the 2>&1 and the redirect matters. The 2>&1 has to be last.
SC2070	-n doesn't work with unquoted arguments. Quote or use [[ ]].
SC2071	> is for string comparisons. Use -gt instead.
SC2072	Decimals are not supported. Either use integers only, or use bc or awk to compare.
SC2076	Don't quote rhs of =~, it'll match literally rather than as a regex.
SC2077	You need spaces around the comparison operator.
SC2082	To expand via indirection, use name="foo$n"; echo "${!name}".
SC2084	Remove '$' or use '_=$((expr))' to avoid executing output.
SC2086	Double quote to prevent globbing and word splitting.
SC2087	Quote 'EOF' to make here document expansions happen on the server side rather than on the client.
SC2088	Tilde does not expand in quotes. Use $HOME.
SC2089	Quotes/backslashes will be treated literally. Use an array.
SC2090	Quotes/backslashes in this variable will not be respected.
SC2091	Remove surrounding $() to avoid executing output.
SC2092	Remove backticks to avoid executing output.
SC2094	Make sure not to read and write the same file in the same pipeline.
SC2096	On most OS, shebangs can only specify a single parameter.
SC2097	This assignment is only seen by the forked process.
SC2098	This expansion will not see the mentioned assignment.
SC2101	Named class needs outer [], e.g. [[:digit:]].
SC2103	Use a (subshell) to avoid having to cd back.
SC2105	break is only valid in loops
SC2107	Instead of [a && b], use [a] && [b].
SC2108	In [[..]], use && instead of -a.
SC2109	Instead of [a || b], use [a] || [b].
SC2110	In [[..]], use || instead of -o.
SC2114	Warning: deletes a system directory. Use 'rm --' to disable this message.
SC2115	Use "${var:?}" to ensure this never expands to /* .
SC2116	SC2116 Useless echo? Instead of 'cmd $(echo foo)', just use 'cmd foo'.
SC2117	To run commands as another user, use su -c or sudo.
SC2119	Use foo "$@" if function's $1 should mean script's $1.
SC2120	foo references arguments, but none are ever passed.
SC2121	To assign a variable, use just 'var=value', no 'set ..'.
SC2122	>= is not a valid operator. Use '! a < b' instead.
SC2123	PATH is the shell search path. Use another name.
SC2124	Assigning an array to a string! Assign as array, or use * instead of @ to concatenate.
SC2125	Brace expansions and globs are literal in assignments. Quote it or use an array.
SC2126	This word is outside of quotes. Did you intend to 'nest '"'single quotes'"' instead'?
SC2128	Expanding an array without an index only gives the first element.
SC2129	Consider using { cmd1; cmd2; } >> file instead of individual redirects.
SC2130	-eq is for integer comparisons. Use = instead.
SC2139	This expands when defined, not when used. Consider escaping.
SC2140	Word is on the form "A"B"C" (B indicated). Did you mean "ABC" or "A\"B\"C"?
SC2141	Did you mean IFS=$'\t' ?
SC2142	Aliases can't use positional parameters. Use a function.
SC2143	Use grep -q instead of comparing output with [-n ..].
SC2144	-e doesn't work with globs. Use a for loop.
SC2145	Argument mixes string and array. Use * or separate argument.
SC2146	This action ignores everything before the -o. Use \(\) to group.
SC2147	Literal tilde in PATH works poorly across programs.
SC2148	Tips depend on target shell and yours is unknown. Add a shebang.
SC2149	Remove $/${} for numeric index, or escape it for string.
SC2150	-exec does not automatically invoke a shell. Use -exec sh -c .. for that.
SC2151	Only one integer 0-255 can be returned. Use stdout for other data.
SC2152	Can only return 0-255. Other data should be written to stdout.
SC2153	Possible Misspelling: MYVARIABLE may not be assigned, but MY_VARIABLE is.
SC2154	var is referenced but not assigned.
SC2155	Declare and assign separately to avoid masking return values.
SC2156	Injecting filenames is fragile and insecure. Use parameters.
SC2157	Argument to implicit -n is always true due to literal strings.
SC2158	[false] is true. Remove the brackets
SC2159	[0] is true. Use 'false' instead
SC2160	Instead of '[true]', just use 'true'.
SC2161	Instead of '[1]', use 'true'.
SC2162	read without -r mangles backslashes
SC2163	Exporting an expansion rather than a variable.
SC2164	Use cd ... || exit in case cd fails.
SC2165	This nested loop overrides the index variable of its parent.
SC2166	Prefer [p] && [q] as [p -a q] is not well defined.
SC2167	This parent loop has its index variable overridden.
SC2168	'local' is only valid in functions.
SC2169	In dash, [[ ]] is not supported.
SC2170	Numerical -eq does not dereference in [..]. Expand or use string operator.
SC2172	Trapping signals by number is not well defined. Prefer signal names.
SC2173	SIGKILL/SIGSTOP can not be trapped.