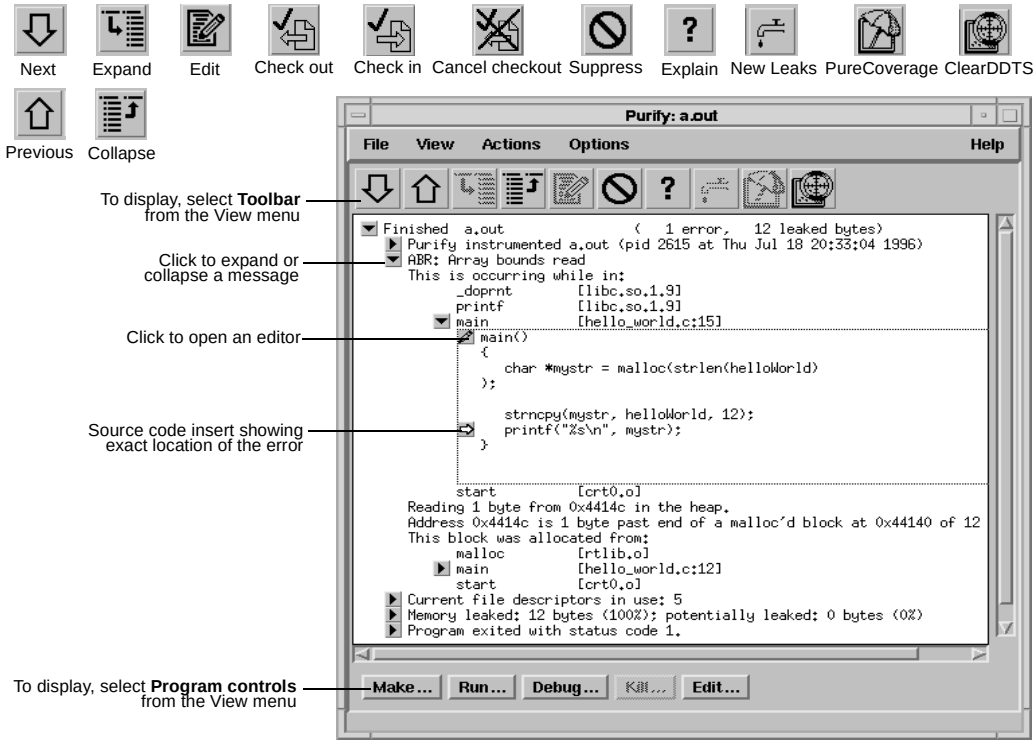
## Using the Purify Viewer

To build a Purify'd program: `% purify cc -g <filename>.o`
Purify opens the Viewer by default when you run a Purify'd program: `% a.out`
To also open a saved view file (`.pv` file) in a Viewer:
`% setenv PURIFYOPTIONS '-view-file=./%v.pv'; a.out; purify -view ./a.out.pv`




### Running a make-run-debug-edit cycle

You can run an entire debugging cycle from the Viewer using the program controls: start a make, run an executable, or launch the debugger or editor.

## Keyboard accelerators

| Key | Action | Menu equivalent |
|---|---|---|
| Control-n, or Down arrow | Move to the next message in the outline hierarchy | Next in the Actions menu |
| Control-p, or Up arrow | Move to the previous message in the outline hierarchy | Previous in the Actions menu |
| Return | Expand the selected message | Expand in the Actions menu |
| DEL | Collapse the selected message | Collapse in the Actions menu |
| Space | Expand if selected message is currently collapsed; Collapse if selected message is expanded | |

# Rational® Purify® Quick Reference

## Purify messages

| Message | Description | Severity* | Message | Description | Severity* |
|---|---|---|---|---|---|
| **ABR** | Array Bounds Read | W | **NPR** | Null Pointer Read | F |
| **ABW** | Array Bounds Write | C | **NPW** | Null Pointer Write | F |
| **BRK** | Misuse of Brk or Sbrk | C | **PAR** | Bad Parameter | W |
| **BSR** | Beyond Stack Read | W | **PLK** | Potential Leak | W |
| **BSW** | Beyond Stack Write | W | **SBR** | Stack Array Bounds Read | W |
| **COR** | Core Dump Imminent | F | **SBW** | Stack Array Bounds Write | C |
| **FIU** | File Descriptors In Use | I | **SIG** | Signal | I |
| **FMM** | Freeing Mismatched Memory | C | **SOF** | Stack Overflow | W |
| **FMR** | Free Memory Read | W | **UMC** | Uninitialized Memory Copy | W |
| **FMW** | Free Memory Write | C | **UMR** | Uninitialized Memory Read | W |
| **FNH** | Freeing Non Heap Memory | C | **WPF** | Watchpoint Free | I |
| **FUM** | Freeing Unallocated Memory | C | **WPM** | Watchpoint Malloc | I |
| **IPR** | Invalid Pointer Read | F | **WPN** | Watchpoint Entry | I |
| **IPW** | Invalid Pointer Write | F | **WPR** | Watchpoint Read | I |
| **MAF** | Malloc Failure | I | **WPW** | Watchpoint Write | I |
| **MIU** | Memory In-Use | I | **WPX** | Watchpoint Exit | I |
| **MLK** | Memory Leak | W | **ZPR** | Zero Page Read | F |
| **MRE** | Malloc Reentrancy Error | C | **ZPW** | Zero Page Write | F |
| **MSE** | Memory Segment Error | W | | | |

** Message severity: F=Fatal, C=Corrupting, W=Warning, I=Informational*

## Suppressing messages

### Message suppression using a `.purify` file

**Suppressing messages from the Viewer:** Click the message, then select **Suppress** from the Options menu. This suppresses messages for the current session. To make the suppression permanent, click **Make permanent** or add the directive shown at the bottom of the suppression dialog to a `.purify` file in one of these standard directories:

- The program directory, to suppress messages from programs in that directory
- Your home directory, to suppress messages from all programs that you run
- The `<purifyhome>` directory, to suppress messages from all programs run by all users at your site

You can also use the `-suppression-filenames` option to specify the filenames of your choice.

### Message suppression directive syntax and examples

**Suppression syntax in a .purify file:** `suppress <message-type> <function-call-chain>`
For `<message-type>`, specify the acronym for the message to be suppressed, wildcard "*" is permitted.
For `<function-call-chain>`, specify a semi-colon delimited chain of call-site specifications each of which may be either a function name or a filename (enclosed in double quotes). Wildcards "*" and "?" are permitted. "..." matches any series of functions.

For example:

- To suppress UMRs from the function `sqrt` add: `suppress umr sqrt`
- To suppress ABRs in any method of class `color` with prefix `test` add: `suppress abr color::test*`
- To suppress all messages from the static and shared versions of `libc` add: `suppress * "libc*"`
- To suppress array bounds messages in all functions called from `main` add: `suppress ab* ...; main`

# Rational® Purify® Quick Reference

## API functions

Include `<purifyhome>/purify.h` in your code and always link with `<purifyhome>/purify_stubs.a`
Useful compile/link options include: `-I`purify -print-home-dir` -L`purify -print-home-dir``

| Commonly used functions | Description |
|---|---|
| **`int purify_describe (char *addr)`** | Prints specific details about memory |
| **`int purify_is_running (void)`** | Returns `"TRUE"` if the program is Purify'd |
| **`int purify_new_inuse (void)`** | Prints a message on all memory newly in use |
| **`int purify_new_leaks (void)`** | Prints a message on all new leaks |
| **`int purify_new_fds_inuse (void)`** | Lists the new open file descriptors |
| **`int purify_printf (char *format, ...)`** | Prints formatted text to the Viewer/log-file |
| **`int purify_watch (char *addr)`** | Watches for memory write, `malloc`, `free` |
| **`int purify_watch_n (char *addr, int size, char *type)`** | Watches memory: type = "`r`", "`w`", "`rw`" |
| **`int purify_watch_info (void)`** | Lists active watchpoints |
| **`int purify_watch_remove (int watchno)`** | Removes a specified watchpoint |
| **`int purify_what_colors (char *addr, int size)`** | Prints color coding of memory |

## Build-time options

Set build-time options on the link line to build Purify'd programs:
`% purify -cache-dir=$HOME/cache -always-use-cache-dir cc ...`

| Commonly used build-time options | Default |
|---|---|
| **`-always-use-cache-dir`** | **`no`** |
| Forces all Purify'd object files to be written to the global cache directory | |
| **`-cache-dir`** | **`<purifyhome>/cache`** |
| Specifies the global directory where Purify caches instrumented object files | |
| **`-collector`** | **not set** |
| Specifies the collect program to handle static constructors (for use with `gcc`, `g++`) | |
| **`-ignore-runtime-environment`** | **`no`** |
| Prevents the run-time Purify environment from overriding the option values used in building the program | |
| **`-linker`** | **system-dependent** |
| Sets the alternative linker to build the executables instead of the system default | |
| **`-print-home-dir`** | |
| Prints the name of the directory where Purify is installed, then exits | |

## Using Purify with other Rational Software products

| Product | Command line syntax |
|---|---|
| PureCoverage | `% purify <purifyoptions> purecov <purecovoptions> cc ...` |
| Quantify | Cannot instrument for Purify and Quantify simultaneously |

# Rational® Purify® Quick Reference

## Run-time options

Set run-time options using the `PURIFYOPTIONS` environment variable:

```
% setenv PURIFYOPTIONS "-log-file=mylog.%v.%p 'printenv PURIFYOPTIONS'"
```

| Commonly used run-time options | Default |
|---|---|
| **`-auto-mount-prefix`**<br>Removes the prefix used by file system auto-mounters | **`/tmp_mnt`** |
| **`-chain-length`**<br>Sets the maximum number of stack frames to print in a report | **`6`** |
| **`-fds-in-use-at-exit`**<br>Specifies that the file descriptor in use message be displayed at program exit | **`yes`** |
| **`-follow-child-processes`**<br>Controls whether Purify monitors child processes in a Purify'd program | **`no`** |
| **`-jit-debug`**<br>Enables just-in-time debugging | **not set** |
| **`-leaks-at-exit`**<br>Reports all leaked memory at program exit | **`yes`** |
| **`-log-file`** †<br>Writes Purify output to a log file instead of the Viewer window | **`stderr`** |
| **`-messages`**<br>Controls display of repeated messages: `"first"`, `"all"` or in a `"batch"` at program exit | **`first`** |
| **`-program-name`**<br>Specifies the full pathname of the Purify'd program if `argv[0]` contains an undesirable or incorrect value | **`argv[0]`** |
| **`-show-directory`**<br>Shows the directory path for each file in the call chain, if the information is available | **`no`** |
| **`-show-pc`**<br>Shows the full pc value in each frame of the call chain | **`no`** |
| **`-show-pc-offset`**<br>Appends a pc-offset to each function name in the call chain | **`no`** |
| **`-view-file`** †<br>Saves Purify output to a view file (`.pv` file) instead of the Viewer. To examine a view file, use `purify -view <filename>.pv` | **not set** |
| **`-user-path`**<br>Specifies a list of directories in which to search for programs and source code | **not set** |
| **`-windows`**<br>Redirects Purify output to `stderr` instead of the Viewer if `-windows=no` | **not set** |

† *Can use conversion characters listed below.*

## Conversion characters for filenames

Use these conversion characters when specifying filenames for options such as `-log-file` and `-view-file`.

| Character | Converts to |
|---|---|
| `%V` | Full pathname of program with "/" replaced by "_" |
| `%v` | Program name |
| `%p` | Process id (pid) |
| qualified filenames (`./%v.pv`) | Absolute or relative to current working directory |
| unqualified filenames (no '/') | Directory containing the program |