# Supported platforms

To run on SDEV

Compile with “nact specs=NX platforms=NXFP2-a32 -d Tests/SystemAbuse” and any other build options you want.

Launch from Visual Studio.

# Initial Setup

By default, abuse expects Abuse.config to be located at \Abuse\Abuse.config on the micro SD card.

This can be changed by passing “-config [configPath]” to the command line.

# What Are tasks?

Tasks are the mini programs that abuse runs. Abuse is capable of simultaneously running a task on every core.

Tasks inherit from BaseTask.h

BaseTask provides some helper functions for logging and creating/destroying threads.

New tasks must implement the following functions:

**InitStatus Initialize(const String& params)**

Called once, when the task runs for the first time.

**StartStatus Start()**

Called every time this gets scheduled to run.

**RunStatus Run()**

Called continuously while this task has time remaining in its slice. This should run as quickly as possible – Abuse cannot reschedule tasks while Run is executing. The default timeslice durations is 100 ms.

**StopStatus Stop()**

Called when this task runs out of time and another task is about to be scheduled.

**ShutdownStatus Shutdown()**

Called once, when the task has been destroyed.

**String GetParamOptions() (Optional)**

Tasks should implement GetParamOptions if they accept arguments in Initialize. GetParamOptions is part of the reply when a list\_available\_tasks command is received. BaseTask::GetParamOptions returns “”.

## To create a new task:

1. Create a class that inherits from BaseTask.
2. Implement the BaseTask functions (Initialize, Start, Run, Stop, Shutdown, (optional)GetParamOptions)
3. Add your task to TaskList.cpp
   1. Include your task
   2. In RegisterTasks: Call Abuse::RegisterTask(“YourTaskName”, BaseTask::CreateTask<YourTask>);
4. Add your task’s .cpps to sdk/Tests/SystemAbuse/!.nact and import your task files into sdk/Tests/SystemAbuse/SystemAbuse.sln
5. To Run your task, use “YourTaskName”. I.e. from AutoAbuse.config call CREATE TASK=YourTaskName

# Abuse.config

If options are unspecified or invalid the default settings are used

Current options are:

**# = Comment. The rest of the line is ignored.**

**WorkingDir=[String workingDir, default=C:\Abuse\]**

Sets the working directory – workingDir is prepended to fileName in Platform::FileOpen(fileName, options)

**ScriptDir=[String scriptDir, default=Scripts\]**

Sets the script directory. Calling FileUtility::LoadScript(scriptName, …) will attempt to open workingDir\scriptDir\scriptName

**StartingScript=[String ScriptName, default=none]**

Attempts to run workingDir\scriptDir\scriptName on startup.

**LogDir=[String logDir, default=Logs\]**

All log files are written to workingDir\logDir\[PlatformName\_N]

PlatformName is the name of the platform (Win32 or SDEV), and N increments from 0 until an unused directory is found.

**T**

**imeStampLogs=[0 or 1, default 0]**

If 1, all Log writes are prepended with a timestamp

**DefaultVerbosity=[0,1,2,3,4, default =4]**

Sets the default verbosity for logs – this is the verbosity level that is used if the LOG\_VERBOSITY argument is not given in the CREATE command.

0: Disables all log writes

1: Enables LogError, other writes disabled

2: Enables LogError and LogWarning

3: Enables LogError, LogWarning, and LogInfo

4: Enables LogError, LogWarning, LogInfo, and LogVerbose

**ForwardToConsole[0 or 1, default=0]**

0: Log writes are only written to file

1: Enabled log writes are also written to the console

**EnableShell=[0 or 1, default=1]**

0: The shell is disabled

1: The shell is enabled

**ShellPort=[0-65535, default=8020]**

Sets the port the shell server will listen on

**TaskRuntimeMillis=[1-unsigned max, default=100]**

Sets the timeslice duration for tasks, in milliseconds.

Each task will run for at least this long before being rescheduled.

# Scripting

Scripting is one way to control which tasks should be run and when they should stop running. The following commands are supported. Parameters in brackets [] are optional.

Whitespace is ignored.

**# = Comment. The rest of the line is ignored.**

**WAIT N**

Wait for N seconds before running the next command.

**STARTLOOP [Label] [Count]**

Runs the commands between this and its paired ENDLOOP.

Label = the name of the iterator variable. It can be used for string substitution in NAME parameters

Count = the number of times to loop. Loops forever is count is unspecified.

**ENDLOOP**

Marks the end of a loop.

**CREATE TASK=TaskType [CORE=int] [NAME=string] [PARAMS=string] [PRIORITY=int] [LOG\_VERBOSITY=int] [LOG\_TIMESTAMPS=0 or 1] [LOG\_TO\_CONSOLE=0 or 1] [SUSPENDED=0 or 1]**

Creates a task of type TaskType. TaskType must match the name of a registered task in TaskList.cpp

CORE=0 to numCores-1

numCores is the number of cores available on the platform.

Specifies which processor the task should run on.

If not specified, the task will run on the core with the fewest tasks assigned to it.

NAME=string, default = “”

The name of the task. Used to identify the task when calling SUSPEND, RESUME, and KILL commands. Must be unique if specified.

When called inside of a loop, can surround loop label in curly braces {} to convert to int.

i.e.

STARTLOOP i 2

CREATE TASK=HelloTask NAME=hello\_{i}

ENDLOOP

Will create two tasks named hello\_1 and hello\_2.

PARAMS=string, default = 16

The PARAMS value is passed directly into the task’s Initialize function.

PRIORITY=0 to 31, default = 16

Determines how frequently a task is chosen. 0 is highest priority, 31 is lowest

The scheduler works by choosing the task with the highest priority, then raising the priority of the other tasks it could have chosen by 1.

LOG\_VERBOSITY=0 to 4, default is set in Abuse.config

Sets the verbosity for this task. Overrides the value in Abuse.config if it is set.

**LOG\_TIMESTAMPS=0 or 1, default is set in Abuse.config**

Sets the timestamp setting for this task. Overrides the setting from Abuse.config if it is set.

0 = Disable timestamps

1 = Enable timestamps

**LOG\_TO\_CONSOLE =0 or 1, default is set in Abuse.config**

Sets the log to console setting for this task. Overrides the setting from Abuse.config if it is set.

0 = Do not write to console

1 = write to console

**SUSPENDED=0 or 1, default=0**

Determines whether the task should be suspended on creation.

0 = The task is not suspended. It can run as soon as it is created.

1 = The task is initially suspended. It will not run until resumed by RESUME or RESUMEALLTASKS

**KILL NAME=taskName**

Kills the specified task. If the task is currently running, Abuse waits for the current call to Run to finish, then calls Stop and Shutdown. Otherwise, Shutdown is called immediately.

taskName = the name of the task to kill. Can use loop label substitution. i.e.

STARTLOOP i 2

CREATE TASK=HelloTask NAME=hello\_{i}

KILL NAME=hello\_{i}

ENDLOOP

Will create hello\_1, kill hello\_1, create hello\_2 and kill hello\_2.

**KILLALLTASKS**

Kills all tasks. This is the only way to kill tasks that were created without NAME.

**SUSPEND NAME=taskName**

Suspends the specified task. If it is currently running, Abuse waits for the current call to Run to finish, then calls Stop. The task will not run again until it is resumed by RESUME=taskName or RESUMEALLTASKS

taskName = the name of the task to suspend. Can use loop label substitution i.e.

STARTLOOP i 2

CREATE TASK=HelloTask NAME=hello\_{i}

SUSPEND NAME=hello\_{i}

ENDLOOP

Will create hello\_1, suspend hello\_1, create hello\_2 and suspend hello\_2.

**SUSPENDALLTASKS**

Suspends all tasks.

**RESUME NAME=taskName**

Allows the specified task to be run again.

taskName = the name of the task to resume. Can use loop label substitution i.e.

taskName = the name of the task to suspend. Can use loop label substitution i.e.

STARTLOOP i 2

CREATE TASK=HelloTask NAME=hello\_{i} SUSPENDED=1

RESUME NAME=hello\_{i}

ENDLOOP

Will create hello\_1, resume hello\_1, create hello\_2 and resume hello\_2.

**RESUMEALLTASKS**

Resumes all tasks.

# Scripting for Unit Tests

Google test unit tests can be run in System Abuse under non-Windows platforms. The \*.nca file is executed, which will run any unit tests included.

* CREATE TASK=UnitTestTask NAME=UnitTestTask\_1 PARAMS=C:\Siglo\system\_abuse\sdk\Tests\Outputs\NX-NXFP2-a32\Tests\testGfx\_SimpleTriangle\Debug\testGfx\_SimpleTriangle.nca SUSPENDED=0 LOG\_VERBOSITY=4

# Python

Abuse can also be controlled by python scripts by using ShellLib.py in Externals\Tests\SystemAbuse\Tools.

All commands (except connect) wait for the command to execute on abuse, then returns a bool saying whether the command was successful.

The commands are:

**connect(ipaddress, port)**

Connects to Abuse. Needs to be called before any other functions.

port should be the same as ShellPort specified in Abuse.config (default=8020)

The ipadress is “192.168.0.10” for SDEV

**stop\_script()**

Stops the currently running script. Abuse cannot run shell commands and a script simultaneously, so this should usually be called before calling any other functions.

**start\_script(filename)**

Attempts to run the script workingDir\scriptDir\filename.

Returns true if the script could be started.

**wait(N)**

waits for N seconds.

**create\_task(taskType, taskName, params, createArgs)**

taskType = the type of task to create. Same as TASK=taskType in script.

taskName = the name of the task. Same as NAME=taskName, except loop name substitution is not supported. Use Python’s string concatenation instead.

params = passed into Initialize, same as PARAMS=params in script

createArgs = instance of TaskCreateArgs. TaskCreateArgs can be used to set:

priority = 0 to 31. Same as PRIORITY in script

core = 0 to numCores - 1. Same as CORE in script

suspended = 0 or 1. same as SUSPENDED in script

verbosity = 0,1,2,3,or 4. Same as LOG\_VERBOSITY in script

timestamps = 0 or 1. Same as LOG\_TIMESTAMPS in script

console\_write = 0 or 1. Same as LOG\_TO\_CONSOLE in script.

**kill\_task(taskName)**

Kills the task with name taskName. Same as KILL NAME=taskName in script.

**kill\_all\_tasks()**

Kills all tasks. Same as KILLALLTASKS in script.

**suspend\_task(taskName)**

Suspends the task with name taskName. Same as SUSPEND NAME=taskName in script.

**suspend\_all\_tasks()**

Suspends all tasks. Same as SUSPENDALLTASKS in script.

**resume\_task(taskName)**

Resumes the task with name taskName. Same as RESUME NAME=taskName in script.

**resume\_all\_tasks()**

Resumes all tasks. Same as RESUMEALLTASKS in script.

**exit\_abuse()**

Attempts to gracefully shutdown the abuse process.