swax / **NAISYS** Public

Code | Issues | Pull requests | Discussions | Actions | Security | Insights

2 Branches 0 Tags | Go to file | Go to file | Code

swax Added an approval agent to the pipeline and reconfigured agents accor…

bc11c36 · yesterday

.vscode	Updated prompt builder to use …	last year
agents	Added an approval agent to the…	yesterday
bin	Added working scdb update dat…	2 weeks ago
eslint-rules	Updated LLM models and prices	5 months ago
fine-tuning	Updated LLM models and prices	5 months ago
src	Added an approval agent to the…	yesterday
.gitignore	Added support for custom env f…	4 days ago
.npmignore	Created fine tuning yaml for ba…	last year
CHANGELOG.md	Added manual and auto comm…	last year
CLAUDE.md	Claude created a tmux script th…	3 weeks ago
CONTRIBUTING.md	Put input command split code i…	last year
LICENSE.md	Improved documentation and a…	last year
README.md	Added complete task command…	2 weeks ago
code-improvement-reco…	Refactor cost tracking to use to…	3 weeks ago
dependency-graph.png	Added sonnet 3.5 model	last year
eslint.config.js	Updated LLM models and prices	5 months ago
naisys-tmux.sh	Claude created a tmux script th…	3 weeks ago
package-lock.json	Separated cost tracking for inp…	3 weeks ago
package.json	Put all naisys tables in same sq…	last week
tsconfig.json	Removed dynamic import from …	last year

About

Command shell driver for LLMs

naisys.org/

#shell #console #automation #ai #autonomy #llm

Readme
MIT license
Activity
50 stars
1 watching
6 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

TypeScript 91.8% **Shell** 6.6% **JavaScript** 1.6%

README | MIT license

NAISYS (Node.js Autonomous Intelligence System)

NAISYS allows any LLM you want to operate a standard Linux shell given your instructions. You can control how much to spend, the maximum number of tokens to use per session, how long to wait between commands, etc.. Between each command NAISYS will wait a few seconds to accept any input you want to put in yourself in case you want to colllaborate with the LLM, give it hints, and/or diagnose the session. Once the LLM reaches the token max you specified for the sesssion it will wrap things up, and start a fresh shell for the LLM to continue on its work.

NAISYS tries to be a minimal wrapper, just helping the LLM operate in the shell 'better'. Making commands 'context friendly'. For instace if a command is long running, NAISYS will interrupt it, show the LLM the current output, and ask the LLM what it wants to do next - wait, kill, or send input. The custom command prompt helps the LLM keep track of its token usage during the session. The 'comment' command helps the LLM think out loud without putting invalid commands into the shell.

Some use cases are building websites, diagnosing a system for security concerns, mapping out the topology of the local network, learning and performing arbitrary tasks, or just plain exploring the limits of autonomy. NAISYS has a built-in system for inter-agent communiation. You can manually startup mulitple instances of NAISYS with different roles, or you can allow agents to start their own sub-agents on demand with instructions defined by the LLM itself!

NPM | Website | Discord | Demo Video

```
npm install -g naisys
```

Node.js is used to create a simple proxy shell environment for the LLM that

- Helps the LLM keep track of its current context size
- Gives the LLM the ability to 'reset' the context and carry over information to a new session/context
- Proxy commands to a real shell, and help guide the LLM to use context friendly commands
- Prevent the context from being polluted by catching common errors like output that includes the command prompt itself
- Allows debugging by way of a 'debug' prompt after each run of the LLM
- A custom 'mail' system for context friendly inter-agent communication
- A browser called 'llmynx' that uses a separate LLM to reduce web page size as well as make links unique across the context
- Cost tracking and cost limits that must be set in the config
- Support for multiple LLM backends, configurable per agent - OpenAI, Google, Anthropic, and self-hosted LLMs

Getting Started

- See notes for Windows users at the bottom of this file
- Install Node.js, NAISYS has been tested with version 20
- Install lynx using `apt install lynx`
- Run `npm install -g naisys`
- Create a `.env` file:

```
# Agent home files and NAISYS specific databases will be stored here
NAISYS_FOLDER="/var/naisys"

# The folder where the website and logs are served from (if not defined then logs put in the naisys folder)
WEBSITE_FOLDER="/var/www"

# Leave api keys/url blank if not using the service
OPENAI_API_KEY="xxx"
GOOGLE_API_KEY="yyy"
ANTHROPIC_API_KEY="zzz"

LOCAL_LLM_URL="http://localhost:1234/v1"
LOCAL_LLM_NAME="minstral instruct v0.2"

# Custom global vars for use in agent configurations here
WEBSITE_URL="http://localhost:8080/"
```

- Create an agent configuration file `smith.yaml`:

```
# Used to identify the agent on the prompt, logs, home dir, mail, etc..
username: smith

# How other agents will understand the role of this agent
title: Software Engineer

# The model to use for console interactions
```

```yaml
# (gpt4turbo, gpt4turbo, gemini-pro, claude3sonnet, claude3opus, local)
shellModel: gpt4turbo

# Only used between sessions to provide guidance for the next session (use a more powerful model for this)
# defaults to the shellModel if omitted
dreamModel: claude3opus

# The model to use for llmynx, pre-processing websites to fit into a smaller context (use a cheaper model)
# defaults to the shellModel if omitted
webModel: claude3haiku

# The model used by the 'genimg' command. If not defined then the genimg command is not available to the LLM
# Valid values: dalle2-256, dalle2-512, dalle2-1024, dalle3-1024, dalle3-1024-HD
imageModel: dalle3-1024

# A system like prompt explaining the agent's role and responsibilities
# You can use config variables in this string
agentPrompt: |
  You are ${agent.username} a ${agent.title} with the job of creating a Neon Genesis Evangelion fan website.
  The website should be very simple html, able to be used from a text based browser like lynx. Pages should be relati
  The location of the website files should be in ${env.WEBSITE_FOLDER}
  The website can be tested with 'llmynx open ${env.WEBSITE_URL}' to see how it looks in a text based browser.
  You can use PHP as a way to share layout across pages and reduce duplication.
  Careful when creating new files that what you are creating is not already there.

# The number of tokens you want to limit a session to, independent of the LLM token max itself
# A lower max relies more on the LLM ending the session with good enough notes to not get lost when the session resta
# A higher max allows the LLM to do more without losing track, but is more expensive
tokenMax: 5000

# The number of seconds to pause after each console interaction for debugging and rate limiting
# No value or zero means wait indefinitely (debug driven)
debugPauseSeconds: 5

# If true, regardless of the debugPauseSeconds, the agent will wake up on messages
# Useful for agents with long debugPauseSeconds, so that they can wake up and reply quickly
wakeOnMessage: false

# The maximum amount to spend on LLM interactions
# Once reached the agent will stop and this value will need to be increased to continue
spendLimitDollars: 2.00

# Command Protection: Useful for agents you want to restrict from modifying the system
#   None: Commands from the LLM run automatically, this is the default setting as well if the value is not set
#   Manual: Every command the LLM wants to run has to be approved [y/n]. Not very autonomous.
#   Auto: All commands are run through the separate LLM instace that will check to see if the command is safe
commandProtection: "none"

# The max number of subagents allowed to be started and managed. Leave out to disable.
# Costs by the subagent are applied to the host agent's spend limit
# Careful: Sub-agents can be chatty, slowing down progress.
subagentMax: 0

# Enable the completetask command for this agent (default: false)
# When enabled, the agent can use completetask to signal task completion, hanging or aborting execution
completeTaskEnabled: true

# Run these commands on session start, in the example below the agent will see how to use mail and a list of other ag
initialCommands:
  - llmail users
  - llmail help
  - cat ${env.NAISYS_FOLDER}/home/${agent.username}/PLAN.md
# Additional custom variables can be defined here and/or in the agent config to be loaded into the agent prompt
```

- Run `naisys <path to yaml or directory>`
  - If a yaml file is passed, naisys will start a single agent
  - If a directory is passed, naisys will start a tmux session with the screen split for each agent

Creating a persistent agent run website (on Digital Ocean for example)

- Create new VM using the LAMP stack droplet template
- Login to the droplet using the web console
- Run `apt install npm`
- Install `nvm` using the `curl` url from these instructions
  - Run `nvm install/use 20` to set node version to 20
- Follow the general install instructions above

Using NAISYS

Testing with tmux (for Development)

For development and testing purposes, you can run NAISYS in a tmux session and send commands to it programmatically:

1. **Use the tmux wrapper script:**

```
# Start the agent in a tmux session
./naisys-tmux.sh start

# Send a talk command and wait for completion
./naisys-tmux.sh talk "create a file called hello.txt with Hello World"

# Send any command manually
./naisys-tmux.sh send "comment 'testing the agent'"
./naisys-tmux.sh trigger  # Send enter to execute

# Get current output
./naisys-tmux.sh output

# Attach to session for manual interaction
./naisys-tmux.sh attach

# Stop the session
./naisys-tmux.sh stop
```

2. **How it works:**
   - The script starts NAISYS in a detached tmux session
   - You can send `talk` commands to communicate with the agent
   - The agent receives messages as if from `admin@naisys`
   - A carriage return triggers the agent to switch from debug to LLM mode
   - The script waits for the command prompt to return, indicating completion
3. **Manual workflow:**

```
# Start session
tmux new-session -d -s naisys-agent 'npm run agent:assistant'

# Send commands
tmux send-keys -t naisys-agent 'talk "your message here"' Enter
tmux send-keys -t naisys-agent Enter  # Trigger agent execution

# View output
tmux capture-pane -t naisys-agent -p
```

This approach is useful for automated testing, CI/CD integration, or when you want to control the agent programmatically while it runs in the background.

The Basics

- NAISYS will start with a debug prompt, this is where you can use and run commands in NAISYS just like the LLM will
- If you hit `Enter` without typing anything, the LLM will run against the prompt
- Afterwards NAISYS will return to the debug prompt

- Depending on how the agents' `debugPauseSeconds` is configured NAISYS will
  - Pause on the debug prompt for that many seconds
  - Pause indefinitely
  - Pause until a new message is received from another agent
- Agents logs are written as html to `{WEBSITE_FOLDER or NAISYS_FOLDER}/logs`

Console Colors Legend

- Purple: Response from LLM, added to context
- White: Generated locally or from a real shell, added to context
- Green: Debug prompt and debug command reponses. Not added to context. Used for diagnostics between calls to LLM
- Red: High level NAISYS errors, not added to the context

Commands

- NAISYS tries to be light, acting as a helpful proxy between the LLM and a real shell, most commands should pass right though to the shell
- Debug Commands
  - `cost` - Prints the current total LLM cost
  - `context` - Prints the current context
  - `exit` - Exits NAISYS. If the LLM tries to use `exit`, it is directed to use `endsession` instead
  - `talk` - Communicate with the local agent to give hints or ask questions (the agent itself does not know about talk and is directed to use `comment` or `llmail` for communication)
- Special Commands usable by the LLM as well as by the debug prompt
  - `comment "<note>"` - The LLM is directed to use this for 'thinking out loud' which avoids 'invalid command' errors
  - `endsession "<note>"` - Clear the context and start a new session.
    - The LLM is directed to track it's context size and to end the session with a note before running over the context limit
  - `pause <seconds>` - Can be used by the debug agent or the LLM to pause execution for a set number of seconds
  - `completetask "<result>"` - Signals that the agent has completed its assigned task
    - For sub-agents: exits the application and returns control to the lead agent
    - For main agents: pauses execution and waits for user input or messages
- NAISYS apps
  - `llmail` - A context friendly 'mail system' used for agent to agent communication
  - `llmynx` - A context friendly wrapping on the lynx browser that can use a separate LLM to reduce the size of a large webpage into something that can fit into the LLM's context
  - `genimg "<description>" <filepath>` - Generates an image with the given description, save at the specified fully qualified path
  - `subagent` - A way for LLMs to start/stop their own sub-agents. Communicating with each other with `llmail`. Set the `subagentMax` in the agent config to enable.

Running NAISYS from Source

Getting started locally

- Install Node.js, NAISYS has been tested with version 20
- Clone the NAISYS repository from Github
- Run `npm install` to install dependencies
- Create a `.env` from the example above
- Run `npm run compile`
- Configure your agent using the examples in the `./agents` folder
- Run `node dist/naisys.js <path to agent yaml file>`

Notes for Windows users

- To use NAISYS on Windows you need to run it locally from source (or from within WSL)
- Use the above instructions to install locally, and then continue with the instructions below
- Install WSL (Windows Subsystem for Linux)

- Install a Linux distribution, Ubuntu can easily be installed from the Microsoft Store
- Make sure to the checked out code perserves the original line endings
  - Files in the /bin folder should have LF endings only, not CRLF
- The `NAISYS_FOLDER` and `WEBSITE_FOLDER` should be set to the WSL path
  - So `C:\var\naisys` should be `/mnt/c/var/naisys` in the `.env` file

Notes for MacOS users

- The browser llmynx requires `timeout` and `lynx`. Run these commands to install them:
  - `brew install coreutils`
  - `brew install lynx`

Using NAISYS for a website

- Many frameworks come with their own dev server
  - PHP for example can start a server with `php -S localhost:8000 -d display_errors=On -d error_reporting=E_ALL`
- Start the server and put the URL in the `.env` file

Changelog

- 1.6: Support for long running shell commands and full screen terminal output
- 1.5: Allow agents to start their own parallel `subagents`
- 1.4: `genimg` command for generating images