Módulo Process

O módulo **process** fornece uma função para execução de comandos do sistema operacional a partir de scripts Lua. Esta função é útil para automação de tarefas administrativas, coleta de informações do sistema e integração com ferramentas externas.

- Execução de comandos com usuário não privilegiado
- Captura de stdout e stderr separadamente
- Retorno estruturado com sucesso/erro
- Suporte a argumentos variáveis

Nota de segurança: Todos os comandos são executados com o usuário `monstasb` para garantir segurança e controle de permissões.

Funções Disponíveis

1. `process.exec(command, [arg1], [arg2], ...)`

Executa um comando do sistema operacional com os argumentos especificados.

Parâmetros:

- **command** (string): Nome do comando/programa a ser executado
- **...** (opcional, strings): Argumentos adicionais para o comando

Retorno:

- **tuple**: `(out, err)` onde:
  - **out** (string ou nil): Saída padrão do comando se bem-sucedido, ou `nil` se falhar
  - **err** (string ou nil): Saída de erro do comando se falhar, ou `nil` se bem-sucedido

Comportamento:

1. O comando é executado com o usuário `monstasb`
2. Se o comando retornar código de saída 0 (sucesso):
   - `out` contém a saída do comando
   - `err` é `nil`

3. Se o comando falhar (código ≠ 0):
   - `out` é nil
   - `err` contém a saída de erro (stderr) do comando
4. Se houver erro na execução (comando não encontrado, etc.):
   - `out` é `nil`
   - `err` contém a mensagem de erro

Exemplo de Uso:

```
-- Executar comando simples
local saida, erro = process.exec("ls", "-la", "/tmp")
if saida then
    print("Conteúdo de /tmp:")
    print(saida)
else
    print("Erro:", erro)
end

-- Executar comando com múltiplos argumentos
local saida, erro = process.exec("df", "-h")
if saida then
    -- Processar saída do df
    local linhas = string.split(saida, "\n")
    for _, linha in ipairs(linhas) do
        print("Linha:", linha)
    end
end

-- Verificar se um serviço está rodando
local saida, erro = process.exec("systemctl", "is-active", "nginx")
if saida then
    local status = string.trim(saida)
    if status == "active" then
        print("Nginx está ativo")
    else
        print("Nginx não está ativo:", status)
    end
else
    print("Erro ao verificar Nginx:", erro)
end
```

```
-- Coletar informações do sistema
local comandos = {
    {"uname", "-a"},
    {"uptime"},
    {"free", "-h"},
    {"df", "-h", "/"}
}

for _, cmd_args in ipairs(comandos) do
    local comando = table.remove(cmd_args, 1)
    local saida, erro = process.exec(comando, table.unpack(cmd_args))
    if saida then
        print("=== " .. comando .. " ===")
        print(saida)
    end
end
```

Informações Adicionais

Execução com Usuário Específico

Todos os comandos são executados com o usuário `monstasb`:

```
-- Segurança: comandos não são executados como root
process.exec("whoami")  -- Retornará "monstasb", não "root"
```

Suporte a Argumentos Variáveis

Aceita qualquer número de argumentos:

```
-- 1 argumento
process.exec("ls")

-- 2 argumentos
process.exec("ls", "-la")
```

```
-- Múltiplos argumentos
process.exec("find", ".", "-name", "*.log", "-type", "f", "-mtime", "+7")
```

Revision #1
Created 3 February 2026 16:44:24 by Marc
Updated 3 February 2026 16:44:35 by Marc