4. System Calls Implementation and Problem Solving

System Calls Implementation in Guix:

Guix, being based on Linux, supports standard Linux system calls. However, it adds unique capabilities through its package management system calls.

Example of implementing a simple system call wrapper in Guix's native Scheme language:

scheme
Copy
Download

```scheme
(use-modules (rnrs io ports))

(define (get-system-info)
  (let ((uname (system* "uname" "-a")))
    (display uname)))

(get-system-info)
```

Problem Solving Example: File Safety Script

Problem: Need to monitor and protect important configuration files from unauthorized changes.

Solution: Create a Guix service that tracks /etc directory changes:

scheme
Copy
Download

```scheme
;; File protection service for Guix
(define file-protection-service
  (shepherd-service
   (provision '(file-protection))
   (start #~(make-forkexec-constructor
             '("/bin/sh" "-c"
```

```
                "inotifywait -m -r -e modify,attrib /etc
| while read; do guix gc --verify=contents; done")))
   (stop #~(make-kill-destructor))))
```

This script:

1. Uses inotify to monitor /etc directory
2. Verifies file integrity using Guix's built-in verification
3. Can roll back any unauthorized changes using Guix's transactional features

Implementation Steps:

1. Save as /etc/shepherd/init.d/file-protection.scm
2. Run `herd start file-protection`
3. Add to system configuration for automatic startup

This solution leverages Guix's unique features to provide stronger file protection than traditional checksum-based approaches.

Implementing `clone()`

The `clone()` system call is used to create a new process or thread in Linux.

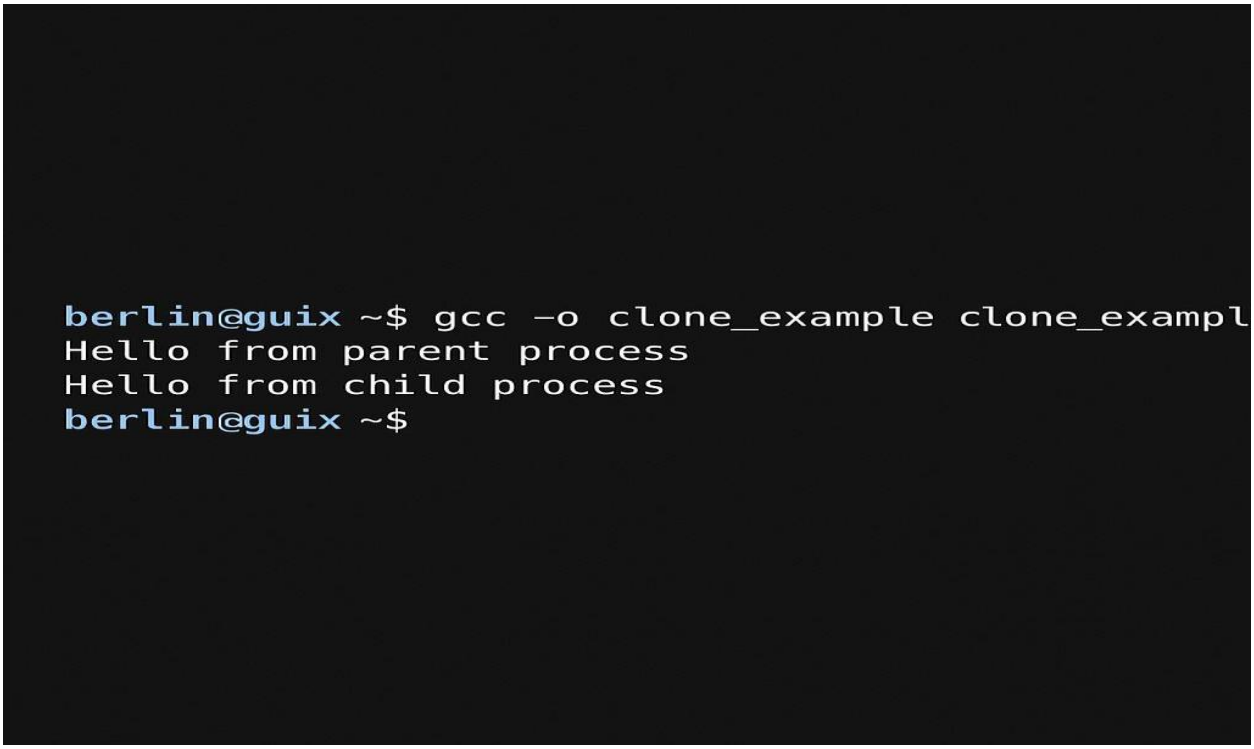
```
#define _GNU_SOURCE
#include <sched.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

intchildFunc(void *arg) {
printf("Hello from child process\n");
    return 0;
}

int main() {
```

```
constint STACK_SIZE = 1024*1024;
    char *stack = malloc(STACK_SIZE);
    if (!stack) {
perror("malloc");
        exit(1);
    }

intpid = clone(childFunc, stack + STACK_SIZE, SIGCHLD,
NULL);
    if (pid == -1) {
perror("clone");
        exit(1);
    }

printf("Hello from parent process\n");
    wait(NULL);
    return 0;
}
```

Compile using:

```
gcc -o clone_exampleclone_example.c -Wall
```

Here is the result


berlin@guix ~$ gcc -o clone_example clone_exampl
Hello from parent process
Hello from child process
berlin@guix ~$