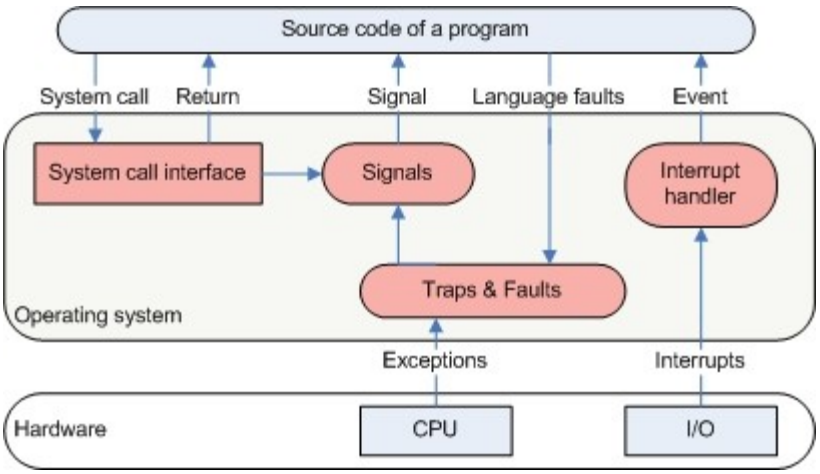
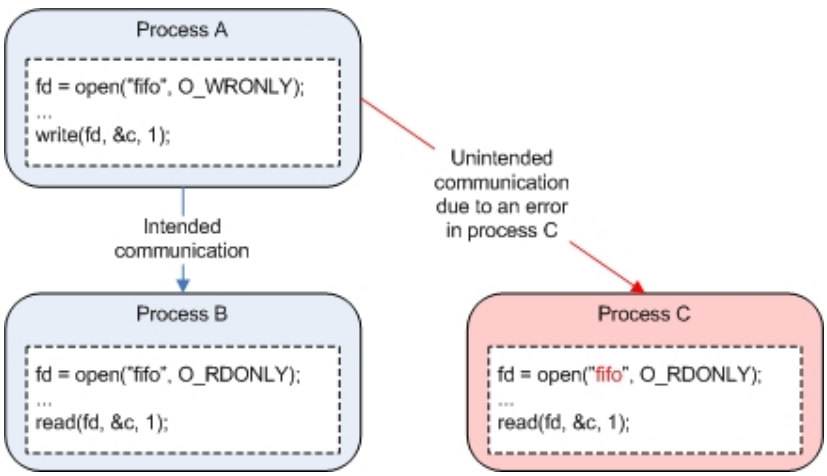
Analysis of Error Propagation

between Software Process in Source Code



Figue 1: Interaction between the operating system and the code of a program



Figue 2: Example of error propagation through an unintended communication channel

Abstract. The purpose of this project was to establish a method that allowed us to study error propagation between software processes. The aim was to determine whether mechanisms for error propagation have characteristics that can be identified in the source code of a program. The focus was on processes running on a single CPU computer with an underlying operating system. As part of the experimental part, C and Linux was chosen as case. Different channels for interaction between user programs and the operating system is illustrated in Figure 1. Linux provides services to user programs through its system call interface. These calls are wrapped in the standard C library and can easily be invoked in code. The report propose a methodology to analyse these using FMEA, and is applied to a subset of system calls related to shared memory. In contrast to general approaches which analyse functionality of software systems, our aim was to identify failure modes caused when a program interacts with these services. This include paths through passed arguments and the return variable. The analysis identified many potential failures, where some could affect other processes and cause error propagation. These failures were then examined to identify their characteristics in code, and a prototype tool was developed to demonstrate that such failures can be detected in source code of a program. The study of how the operating system works and interacts with user programs, was presented at ESREL 2007 [1].

An example of error propagation in software is given using the inter-process communication FIFO (First In First Out), where two processes communicate using the file system. To use this mechanism, the processes have to open the same file identifier, which is a file in the file system (visible to all). Figure 2 illustrates a scenario: process A and B use FIFO for communication using the same file identifier. If another process (process C) by "mistake" makes use of the same file identifier (same filename and path), it would interfere with the communication between process A and B and thus cause error propagation. While the use of FIFOs is an intended communication channel, the error in process C creates a channel between process A and C which was not intended. The result in this case is that process B will not be able to retrieve all data sent from process A because process C reads from the same communication channel. Once a read occurs, the buffer is cleared.

Author:	Supervisor:
Sizarta Sarshar	Rune Winther

Institution:
Østfold University College
Department for Computer Science, Halden

In collaboration with:
Institute for Energy Technology (IFE)
OECD Halden Reactor Project
ICT Risk and Dependability (RID), Halden

References

[1] S. Sarshar, J. E. Simensen, R. Winther, and R. Fredriksen, “Analysis of error propagation mechanisms between software processes,” in *Safety and Reliability Conference*, Stavanger, Norway, June 25-27, 2007, pp. 91–98.