PROCESS MODELING IN THE LINUX OPERATING SYSTEM

Valentina Kukenska[1], Matyo Dinev[1], Petar Minev[1], Ilian Varbov[1]
[1]Technical University of Gabrovo

Abstract
This paper focuses on the processes in the LINUX operating system. It looks at their possible states. The study proposes a probability analytical model based on Markov chains. The model can be used to define the state of a given process in a specific moment of time, and to evaluate the probability of the process entering a certain state.

Keywords: Process, Linux, modeling, probability analytical model, General Purpose Simulation System (GPSS)

1. INTRODUCTION

Different operating systems use different concepts: job, task, process. The term process was first used in the MIT MULTICS operating system of MIT in the 1960s and dominates today's operating systems. The shortest definition of a process is a program in the course of its implementation.

Modern computer systems can perform multiple processes simultaneously. It is accepted that this parallelism is called multiprogramming. Its purpose is more efficient use of the resources of the computer system. There are many processes at the same time, and the central processor (CPU) is only one and can perform only one of these processes at any one time. This process is running. The other processes are in some other state.

This paper focuses on the state of the LINUX operating system processes. A probabilistic analytical model based on Markov chains is proposed. It allows you to determine the probability of a process entering a certain state.

2. STATE OF THE PROCESSES IN LINUX

The LINUX processes can be found in one of the states (1 ÷ 6) represented by the state diagram of Fig. 1.

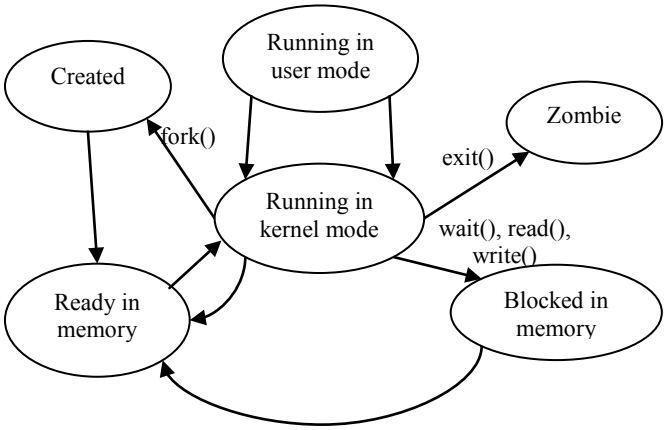

Fig. 1. *Diagram of process states*

1. Created. This is the initial state in which the process enters the system. It is a transient state of the process. It is almost created but not yet fully operational.
2. Ready in memory. The process is ready for execution and is in the memory.
3. Running in kernel mode. The process is executed in kernel mode, with the CPU executing commands from the kernel.
4. Running in user mode. The CPU executes commands from the process related user program.
5. Blocked in memory. The process awaits the occurrence of an event and is in the memory.
6. Completion (zombie). This is the final state of each process.

As shown in Figure 1, the execution state is split into two: execution in kernel mode and execution in user mode. When the process is in a state of execution in user mode, user

instructions are executed. In the kernel mode execution state, kernel instructions are performed in the context of the current process. The process can not directly go from user mode to blocking, standby, or completion. Such transitions are only possible through the "running in kernel mode" intermediate state. It is not possible to switch from "ready in memory" directly to "execution in user mode".

3. ANALYTICAL MODEL

Each process from its creation to its completion passes through different states (Figure 1). Process states are a finite number, but transition from one state to another changes over time and can be seen as a random process with discrete states. This process can be presented with a graph of states. Each state is marked with a circle, and any possible transition from state to arrow state.

Each possible sequence of process states forms a chain. Since the number of states is limited, the chain is final. Status transitions in state are performed at unknown time points, therefore the chain has a continuous time.

Let us present with a graph the state of the processes in the LINUX OS (Fig. 2).

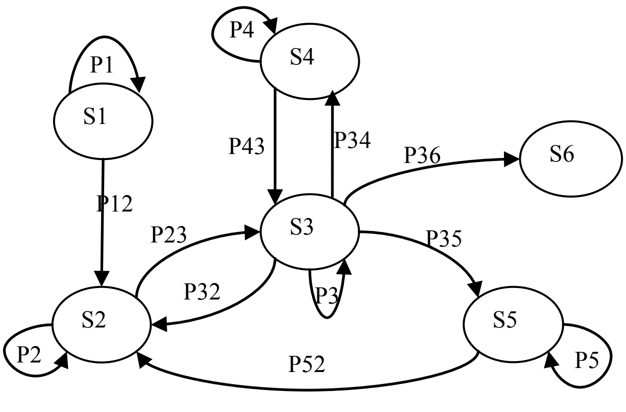

Fig. 2. *Graph of the process states*

We assume that the event $S_1(t)$ corresponds to the process creation $S_2(t)$ - the state of ready, $S_3(t)$ - of the state performed in the kernel mode, $S_4(t)$ - the state in the user mode, $S_5(t)$ - the state blocked and $S_6(t)$ - of the completed process.

Every process is always in one of the states at a time. Therefore, only one of the events $S(t) = S_i$, $(i = 1, 2, ..., n)$ occurs at any moment of time. If $p_i(t)$ indicates the probability of the process being in the S_i state, the normative condition (1)

$$\sum_{i=1}^{n} p_i(t) \qquad (1)$$

For every two time points, it is likely that the process will go into another state, as well as maintain its current state. These probabilities can be considered conditional.

The initial condition of the process is set with the vector of the initial probability distribution of the states (2):

$$P(0)= |p_1(0), p_2(0), \dots , p_n(0)| = |p_i(0)|_{1\times n} \quad (2)$$

The vector $P(0)$ can be set in advance. It can also be occasional for a given time.

The state of the process after a number of times of change of time can be presented as a distribution of the probabilities of the states (3):

$$p(k)= |p_1(k), p_2(k), \dots , p_n(k)| = |p_i(k)|_{1\times n}, \quad (3)$$

Where $k=1,2,3,...$ and $0\leq p_i(k)\leq 1$. The transition probabilities $P_{ij}(k)$ for passing the process from state S_i to state S_j are set with an index matrix of transient states $P=|P_{ij}(k)|_{n\times n}$, $i,j=1,2, \dots , n$.

With a given vector of the initial probability distribution of states $P(0)$ and a known matrix of transition probabilities P, the probability distribution vector of the states $p(k)$ for each step k can be determined.

If before step k, i.e. in the $(k-1)$ step the process is probable $p_i(k-1)$ in the S_i state, then the probability that after the step k is in the state S_j is determined by the full probability formula (4):

$$p_j(k) = \sum_{i=1}^{n} p_i(k-1).p_{ij} \qquad (4)$$

where $j = 1,2, ..., n$, and $p_i(k-1)$ are the probabilities of the hypotheses for conditions S_i $(i = 1,2, ..., n)$, and $P_{ij}(i=1,2, ..., n)$ are the probability probabilities given in j - pillar of the matrix P.

The analytical model of the process can be presented with the following equations and dependencies:

$$\begin{cases} p_1 = p_{11}.p_1 + p_{21}.p_2 + \cdots + p_{n1}.p_n \\ p_2 = p_{12}.p_1 + p_{22}.p_2 + \cdots + p_{n2}.p_n \\ \dots \\ p_n = p_{1n}.p_1 + p_{2n}.p_2 + \cdots + p_{nn}.p_n \\ p_1 + p_2 + \cdots + p_n = 1 \end{cases} \qquad (3)$$

If the probability matrix of the transient states is known, it is possible to determine the

probabilities of the process incident in each of the possible S_i states.

If the probability matrix of the transient states is:

$$P = \begin{bmatrix} 0,1 & 0,9 & 0 & 0 & 0 \\ 0 & 0,8 & 0,2 & 0 & 0 \\ 0,1 & 0,2 & 0,3 & 0,3 & 0,1 \\ 0 & 0 & 0,4 & 0,6 & 0 \\ 0 & 0,3 & 0 & 0 & 0,7 \end{bmatrix}$$

the following result is obtained: P_1=0,02; P_2=0,48; P_3=0,24; P_4=0,18 and P_5=0,08.

4. MODEL SIMULATION

The proposed model is described in process-oriented language and simulated with the General Purpose Simulation System (GPSS).

```
; MS/1/
        GENERATE        90
PS1     QUEUE           QSS1
        SEIZE           SS1
        DEPART          QSS1
        ADVANCE         1
        RELEASE         SS1
        TRANSFER        .50,,PS1
PS2     QUEUE           QSS2
        SEIZE           SS2
        DEPART          QSS2
        ADVANCE         1
        RELEASE         SS2
        TRANSFER        .50,,PS2
PS3     QUEUE           QSS3
        SEIZE           SS3
        DEPART          QSS3
        ADVANCE         1
        RELEASE         SS3
        TRANSFER        .17,,PS1
        TRANSFER        .17,,PS2
        TRANSFER        .15,,PS3
        TRANSFER        .17,,PS5
        TRANSFER        .17,,PS6
PS4     QUEUE           QSS4
        SEIZE           SS4
        DEPART          QSS4
        ADVANCE         1
        RELEASE         SS4
        TRANSFER        .50,,PS3
        TRANSFER        .50,,PS4
PS5     QUEUE           QSS5
        SEIZE           SS5
        DEPART          QSS5
        ADVANCE         1
        RELEASE         SS5
        TRANSFER        .50,,PS2
        TRANSFER        .50,,PS5
PS6     QUEUE           QSS6
        SEIZE           SS6
        DEPART          QSS6
        ADVANCE         1
        RELEASE         SS6
        TERMINATE
; MS/2/
        GENERATE        100
        TERMINATE       1
```

Simulation of the model has given the status of the processes for a given time interval.

When looking at a process, part of the results obtained are:

FACILITY	ENTR.	UTIL.	AVE.TIME	AVAIL.	OWNER
SS1	2	0.010	0.500	1	1
SS2	4	0.040	1.000	1	0
SS3	2	0.020	1.000	1	0
SS4	2	0.020	1.000	1	0
SS5	1	0.010	1.000	1	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.
QSS1	1	0	2	2	0.000
QSS2	1	0	4	4	0.000
QSS3	1	0	2	2	0.000
QSS4	1	0	2	2	0.000
QSS5	1	0	1	1	0.000

For the specified time interval, the process under investigation was two times S_1, four times S_2, two times S_3 and S_4, one time S_5. At the end of the interval is the execution state (S_1).

The following data is obtained from ten processes:

FACILITY	ENTRIES	UTIL.	AVE.TIME	AVAIL.	OWNER
SS1	23	0.220	0.957	1	11
SS2	29	0.290	1.000	1	9
SS3	27	0.270	1.000	1	0
SS4	17	0.170	1.000	1	0
SS5	8	0.080	1.000	1	0
SS6	7	0.070	1.000	1	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME
QSS1	1	0	23	22	0.010	0.043
QSS2	2	1	30	28	0.020	0.067
QSS3	2	0	27	24	0.030	0.111
QSS4	1	0	17	17	0.000	0.000
QSS5	1	0	8	8	0.000	0.000
QSS6	1	0	7	7	0.000	0.000

5. CONCLUSION

A probabilistic analytical model of processes in the LINUX operating system is proposed. It determines the probability that the process being studied is in a certain state at a given time, as well as the probability of it being in a certain state after a certain time. The model is simulated with the GPSS software. The proposed model can be used to model and study discrete-state processes.

6. ACKNOWLEDGMENT

The work presented in this paper was funded been prepared with the financial assistance of contract No. 1907E for conducting research on a project "Implementation of innovative ICT technologies in training" at the Technical University of Gabrovo.

7. REFERENCES

[1] Clymer, J. System Analysis Using Simulation and Markov Models, Prentice-Hall, N. J., 1990.

[2] Daniel P Bolvet, M Cesati, Understanding the LINUX Kernel, O'Reilly Media, 2000

[3] FerreiraF., A. Pacheco, Simulation of semi-Markov processes and Markov chains ordered in level crossing, Next Generation Internet Networks, 2005, pp. 121-128.Tavel, P. 2007. Modeling and Simulation Design. AK Peters Ltd., Natick, MA.

[4] Herbort C., UNIX- Practical Visual Guide, SoftPress, 2005

[5] Silberschatz A., Operating System Concepts, N.Y., 2009

[6] Stallings W., Operating Systems: Internals and Design Principles, N.Y., 2008.