Relatório do EP1 de Sistemas Operacionais

MAC422

Professor: Alan Durham

Nomes:

- Karina Suemi Awoki nºUSP: 7572102
- Ruan de Menezes Costa nºUSP: 7990929

CHAMADAS DE SISTEMA:

1 RodeVeja

Essa chamada ao sistema cria um processo filho, e faz com que o processo atual (pai) seja interrompido e só volte a ser executado após a finalização de seu filho.

Para isso usamos as funções *fork()* e *execve()* da biblioteca *unistd.h* e *waitpid()* da biblioteca *sys/wait.h*
Primeiramente clonamos o processo, que está sendo executado no momento, com a função *fork*.

- Para o processo com *PID* != 0 (que é o processo atual), fazemos com que este seja interrompido até que o processo filho seja finalizado.

  Isso é feito através da função *waitpid()* que recebe como parâmetro o *PID* do filho a quem ele irá esperar para continuar.

- Já para o processo com *PID* = 0 (que é o processo filho), fazemos com que ele assuma o valor do programa passado pela entrada padrão e o executamos.

  Isso tudo é feito através da função *execve()* onde passamos como parametro o nome do arquivo e os seus respectivos argumentos (que armazenamos em *args*). Isso é feitp para que o processo filho (que quando foi clonado possuia as características do pai) possa se transformar no programa que foi passado pela entrada padrão, e assim ser executado (com os argumentos que também foram passados pela entrada padrão).

2 Rode

Essa chamada ao sistema clona o processo atual com a função *fork()*, criando um processo filho e, ao verificar o processo filho (quando *PID* = 0), o executa passando o arquivo vindo da entrada padrão juntamente com seus argumentos, para que a processo filho se tranforme no arquivo passado.

Observa-se que não usamos a função *waitpid* como na chamada ao sistema anterior, isso ocorre porque em *rode* queremos executar o processo em backgroud, ou seja, queremos executar o filho juntamente com o pai. Logo, não é necessário que façamos com que o pai espere o filho ser finalizado para continuar executando suas atividades.

3 ProtegePraCaramba

Para mudar a proteção do arquivo passado por entrada padrão, foi usado o comando *chmod*, alterando a permissão de acesso do arquivo para (000). Este comando vem da biblioteca *sys/stat.h*

- usuário: permissão negada para *leitura, gravação* e *execução*.
  0 = (000)2
- grupo: permissão negada para *leitura, gravação* e *execução*.
  0 = (000)2
- outros: permissão negada para *leitura, gravação* e *execução*.
  0 = (000)2

4 LiberaGeral

Para mudar a proteção do arquivo passado por entrada padrão para (777), também foi usado o comando *chmod* através da biblioteca *sys/stat.h*

- usuário: permissão concedida para *leitura, gravação* e *execução*.
  7 = (111)2
- grupo: permissão concedida para *leitura, gravação* e *execução*.
  7 = (111)2
- outros: permissão concedida para *leitura, gravação* e *execução*.
  7 = (111)2