Renan Gonçalves.
BEC

System calls -

execve - executa o programa

brk, sbrk - altera o tamanho do segmento de dados;
brk() define o final do segmento de dados para o valor especificado por *addr* (***int brk (void** * addr* **))** quando esse valor for razoável, o sistema tem memória suficiente e o processo não excede seu tamanho máximo de dados.

access, faccessat - verifica as permissões do usuário para um arquivo.

- *Valor de retorno*: Em caso de sucesso (todas as permissões solicitadas concedidas ou o *modo* é **F_OK** e o arquivo existe), zero é retornado. Em caso de erro (pelo menos um bit em *o modo* pediu uma permissão negada ou o *modo* é **F_OK** e o arquivo não existe ou ocorreu algum outro erro), -1 é retornado,e *errno* é definido apropriadamente.
- **ENOENT :** Um componente do *pathname* não existe ou está pendente link simbólico.

open, openat, creat - abrir e possivelmente criar um arquivo.

- openat()

A chamada de sistema openat () opera exatamente da mesma maneira que open (), exceto para as diferenças descritas aqui.

Se o caminho fornecido no caminho for relativo, ele é interpretado relativo ao diretório referido pelo descritor de arquivo dirfd (em vez de relativo ao diretório de trabalho atual da chamada processo, como é feito por open () para um nome de caminho relativo).

Se o nome do caminho for relativo e dirfd for o valor especial AT_FDCWD, então
o nome do caminho é interpretado em relação ao diretório de trabalho atual de
o processo de chamada (como open ()).

Se o nome do caminho for absoluto, dirfd será ignorado.

stat, fstat, lstat, fstatat - pega o status do arquivo.

- **fstat** () é idêntico a **stat** (), exceto que o arquivo sobre o qual a informação a ser recuperada é especificada pelo descritor de arquivo *fd*.
- **Valor de retorno**: em caso de sucesso, zero é retornado, em caso de erro, -1 é retornado e *errno* é definido apropriadamente.

mmap, munmap - mapear ou desmapear arquivos ou dispositivos na memória.

void * mmap (void * *addr* **, size_t** *length* **, int** *prot* **, int** *flags* **, int** *fd* **, off_t** *offset* **);**

cria um novo mapeamento no espaço de endereço virtual do processo de chamada. O endereço inicial para o novo mapeamento é especificado em *addr* . O argumento *comprimento* especifica o comprimento do mapeamento (que deve ser maior que 0).

Se *addr* for NULL, então o kernel escolhe o endereço (alinhado à página)no qual criar o mapeamento.

- **Valor de retorno**: Em caso de sucesso, **mmap** () retorna um ponteiro para a área mapeada. Em caso de erro, o valor **MAP_FAILED** (ou seja, *(void *) -1*) é retornado, e *errno* é definido para indicar a causa do erro.

close – fecha um descrito de arquivo.

int close(int *fd***)**

close () fecha um descritor de arquivo, de forma que ele não se refira mais a nenhum arquivo e possa ser reutilizado. Todos os bloqueios de registro (consulte fcntl (2)) mantidos no arquivo ao qual ele estava associado e pertencentes ao processo são removidos (independentemente do descritor de arquivo usado para obter o bloqueio).

valor de retorno: retorna zero em caso de sucesso. Em caso de erro, -1 é retornado e *errno* é definido apropriadamente.

read – ler um descritor de arquivo.

ssize_t read (int *fd* **, void *** *buf* **, size_t** *count* **);**

read () tenta ler até *contar* bytes do descritor de arquivo *fd* para o buffer começando em *buf.*

- **Valor de retorno:** Em caso de sucesso, o número de bytes lidos é retornado (zero indica fim do arquivo), e a posição do arquivo é avançada por este número. Isto não é um erro se este número for menor que o número de bytes Requeridos.

mprotect, pkey_mprotect - define uma proteção em uma região da memória.

int mprotect(void *addr, size_t len, int prot);

mprotect () muda as proteções de acesso para as páginas de memória do processo de chamada contendo qualquer parte do intervalo de endereços no intervalo [*addr* , *addr* + *len* - 1]. *addr* deve ser alinhado a um limite de página.

- **Valor de retorno:** On success, **mprotect**() and **pkey_mprotect**() return zero. On error, these system calls return -1, and *errno* is set appropriately.

write - escreve em um arquivo descritor.

ssize_t write(int *fd*, const void **buf*, size_t *count*);

write () escreve para *contar* bytes do buffer começando em *buf* para o arquivo referido pelo descritor de arquivo *fd*.

- **Valor de retorno:** Em caso de sucesso, o número de bytes gravados é retornado. Em caso de erro, -1 é retornado e *errno* é definido para indicar a causa do erro.

arch_prctl - define o estado do thread específico da arquitetura

int arch_prctl (*código* int , sem sinal longo * *addr*);

arch_prctl () define o processo específico da arquitetura ou o estado da thread.*o código* seleciona uma subfunção e passa o argumento *addr* para ela; *addr* é interpretado como um *longo sem sinal* para as operações de "conjunto", ou como um *longo sem sinal* * , para as operações "get".

- **valor de retorno:** Em caso de sucesso, **arch_prctl** () retorna 0; em caso de erro, -1 é retornado e *errno* é definido para indicar o erro.

uname - obtém o nome e informações sobre o kernel atual.

int uname (struct utsname * *buf* **);**

uname () retorna informações do sistema na estrutura apontada por *buf* . A estrutura *utsname* é definida em *<sys / utsname.h>* :

valor de retorno: Em caso de sucesso, zero é retornado. Em caso de erro, -1 é retornado e *errno* é definido apropriadamente.

exit_group - sai de todos os tópicos em um processo.

void exit_group (*status* **interno);**

Esta chamada de sistema é equivalente a _exit (2), exceto que ela termina não apenas o thread de chamada, mas todos os threads no processo de chamada grupo de discussão.

- **Valor de retorno**: Esta chamada de sistema não retorna.