NOVA buffer manager for SDC portable DAQ

Users manual

Masaharu NOMACHI
National Laboraroty for High Energy Physics (KEK)

Y. TAKEUCHI
Tokyo Inst. of Tech.

Introduction

A data buffer management takes one of the most important roll in a data acquistion system. NOVA has been developed originaly on the OS9 operating system for a balloon experiment.

Processes

NOVA buffer manager consists with three kinds of processes. One is a daemon process called novad. Second one is a buffer manager called bufmgr. The last one is watch dog process for each user processes. The daemon waits socket connection from user processes. A user process send its process id etc. to the daemon. The daemon forward them to the buffer manager. The daemon receives the informations for communication with the buffer manager and return them to the user process. Then, the user process starts communicate with the buffer manager with the informations. Its communication is done with UNIX message queue. These sequence of installation is done with a function nova open.

Process link

Consumer processes participating the buffer manager system are connected to one line. There is no branching nor any conversing within the line. A process with higher priority comes at the upstream. A lower priority process lines at the downstream. The buffers are forwarded from a upstream process to a down stream process. The order of the buffers are not changed. Lower priority processes may not get all buffer.

Priority and significance

Data flow is controled with the process priority and the buffer significance. A buffer is charactrized by its significance. A buffer with bigger significance is always deliverd. The control is done by a sum of the buffer significance and the process priority. A buffer with "significance" is always delivered to a process with "priority", if those values satisfy the following condition.

$$significance + priority > threshold$$

A "threshold" is 100 in NOVA buffer manager. The figure shows this condition. A buffer which is not deliverd is returned to free buffer queue. It is not forwarded to the downstream processes. It is to keep the order of buffers. The down stream process may get a fraction of buffer but the order is same as original order. It may happen only if the producers can not get enough number of free buffers. When the producers do not request the free buffer, it may not happen.

User functions

Nova descriptor

A process opens a file with file descriptor. On the similler way, A process can open a path to the buffer manager with a nova descriptor. A process can open more than two processes with using individual nova descriptors. Its structure is defined in the header file "novad.h". A nova_open function creates the descriptor area and returns its address.

Name

It is possible to start more than one buffer manager within one UNIX system. Each buffer manager is characterized by a "NAME". It is a name of socket within a system, which is a file name. It is recommended to have an absolute path name at "/tmp" device.

Buffer size

Producers can define the buffer size, which should be less than the maximum size. The consumers can receive its size. However, it does not mean the buffer is managed as valiable size buffer. It is managed as a fixed length buffer. Therefore, the number of buffer does not change.

request

The buffer manager assign "request" buffers to a process before it returns the buffer. It is recommended that "$request = 1$"in usual applications. The process with "$request = 0$" gets a buffer on demand. The buffer never forwarded to the downstream processes. Therefore, one should not use this mechanism for the process with higher priorities. This mechanism is designed for the process with lower priorities like "$priority = 1$" processes. The processes with higher priority can get a buffer on demand with "pause", "resume" functions.

pause and resume

A process can “pause” the request of buffer. Buffers are not deliverd to “pause” process, and are forwarded to the down stream processes. The buffers deliverd to the process are forwarded on the execution of the “pause” function. A process can hold the buffer delivered before it pauses. It is a function “pause_hold()”. The buffers HOLD will be returned to free buffer pool on “put”. Those are not deliverd to the process on its down stream. The process can “resume” to “get” a buffer. All buffers “hold”ed by the process is returned to the free buffer pool on execution of the “resume” function. A sequence of “resume”, “get”, “pause” works as “get on demand” but the buffer is forwarded to the downstream. It may stop the flow if the process has very high priority. The “pause”d process has a zero request. Therefore “get” on “pause” works also as “get on demand” but the buffer is returned to the free buffer pool.

nd = nova_open(name,priority,Nrequest)

char *name;	Path name to the buffer manager.
int priority;	Process priority
int Nrequest;	A number of request
struct NOVAD *nd;	Pointer to the nova descriptor.

nova_close(nd)

struct NOVAD *nd;	Pointer to the nova descriptor.

nova_get(nd,address,length)

struct NOVAD *nd;	Pointer to the nova descriptor.
char **address;	pointer to buffer address pointer.
int *length;	pointer to buffer size.

nova_get_nowait(nd,address,length)

struct NOVAD *nd;	Pointer to the nova descriptor.
char **address;	pointer to buffer address pointer.
int *length;	pointer to buffer size.

nova_put(nd,address,size)

struct NOVAD *nd;	Pointer to the nova descriptor.
char *address;	buffer address pointer.
int length;	buffer size.

nova_pause(nd)

struct NOVAD *nd;	Pointer to the nova descriptor.

nova_pause_hold(nd)

struct NOVAD *nd;	Pointer to the nova descriptor.

nova_resume(nd,Nrequest)

struct NOVAD *nd;	Pointer to the nova descriptor.
int Nrequest;	A number of request

Startup and termination

Utilities